Documentation de l'interface du logiciel OCAPI : Version 2.0

Sabine Moisan, Florimond Ployette, Monique Thonnat, Véronique Clément, Régis Vincent

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Documentation de l'Interface du Logiciel OCAPI Version 2.0

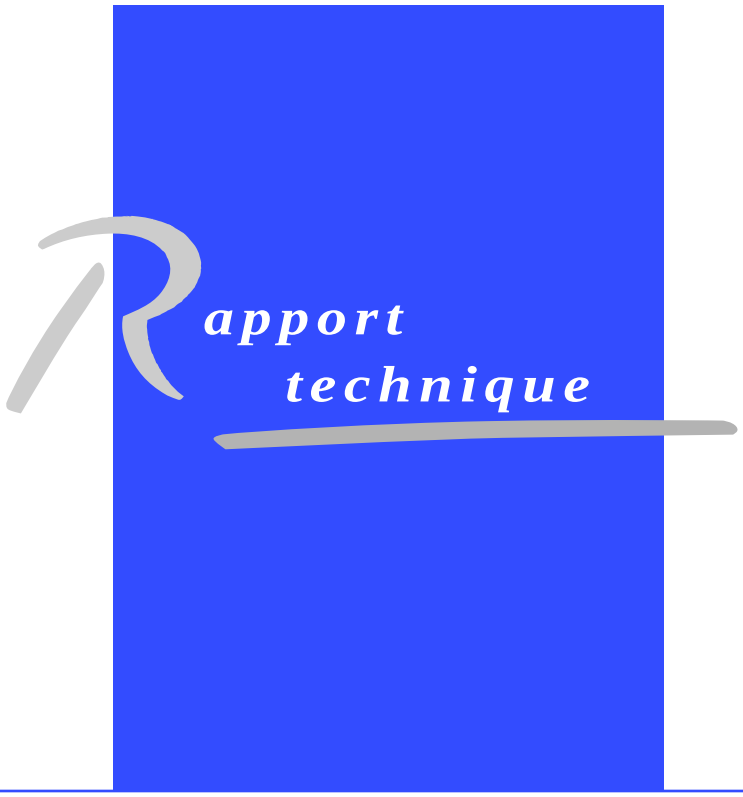
Sabine Moisan, Florimont Ployette, Monique Thonnat,

Véronique Clément, Régis Vincent

N° 0186

Décembre 1995

PROGRAMME 3



ISSN 0249-6399

Documentation de l'Interface du Logiciel OCAPI Version 2.0

Sabine Moisan, Florimont Ployette, Monique Thonnat,
Véronique Clément, Régis Vincent

Programme 3 — Intelligence artificielle, systèmes cognitifs
et interaction homme-machine
Projet Orion

Rapport technique n ° 0186 — Décembre 1995 — 71 pages

Résumé : Ce document est composé de deux parties : la première, intitulée **Présentation générale**, décrit succinctement les interfaces graphiques disponibles autour d'OCAPI ; la seconde, intitulée **Menus et dispositifs de cliquage de l'interface**, présente de façon détaillée les différents menus et boutons de cette interface.

Le manuel de référence du logiciel est disponible, en rapport technique n° RT-0183 [MVT+95]

Mots-clé : pilotage de programmes, système à base de connaissances, interface graphique

(Abstract: pto)

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

OCAPI Graphical User Interface Manual
Version 2.0

Abstract: This document is composed of two parts: the first one entitled **Présentation générale**, shortly describes the graphical interface available in OCAPI. The second part: **Menus et dispositifs de cliquage de l'interface** focuses on all popup menus and boutons of the GUI of OCAPI.

The reference manual of OCAPI is also a technical report (n° RT-0183 [MVT+95])

Key-words: program supervision, knowledge-based system, graphical interface

1 Présentation générale

Afin de faciliter l'utilisation d'OCAPI, des interfaces graphiques ont été développées. Ces interfaces fonctionnent par menus et utilisent le multi-fenêtrage. Les intitulés des menus et des fenêtres sont soit en anglais soit en français, selon la version choisie au moment de la livraison d'OCAPI. Dans la suite de ce document ces intitulés seront donnés en français. Trois menus principaux sont accessibles au lancement d'OCAPI: le menu de développement d'une base de connaissance [**Developpement**], le menu d'utilisation du système expert [**Utilisation**] et le menu des réglages des diverses variables [**Reglage**]. Pour un système expert donné, il est toujours possible de définir une interface particulière (en utilisant le langage de description d'interfaces Aïda).

1.1 Menu de développement [Developpement]

Ce menu s'adresse à l'expert en cours de développement d'une base de connaissance et offre les fonctions suivantes :

– Gestion d'informations générales sur chacune des différentes bases de connaissance : nom, objet, auteur, localisation, requêtes initiales, fonctionnalités... (figure 1).

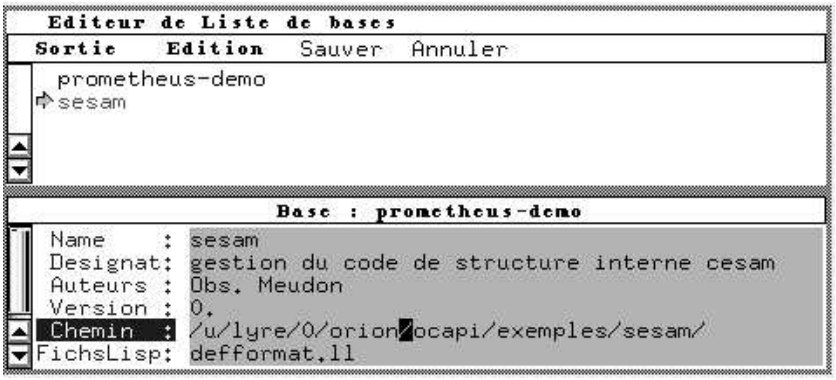


FIG. 1 – *Editeur de bases de connaissance (informations générales)*

– Visualisation d'une base de connaissance : une visualisation globale de la base est possible soit sous forme textuelle, soit sous forme arborescente. La forme textuelle (figure 2) montre l'ensemble des entités de la base de connaissance

dans un tableau à quatre composants : les buts, les opérateurs, les règles associées aux buts et les règles associées aux opérateurs. Il est possible de visualiser les dépendances entre ces différentes entités et d'accèder à leurs éditeurs spécifiques.

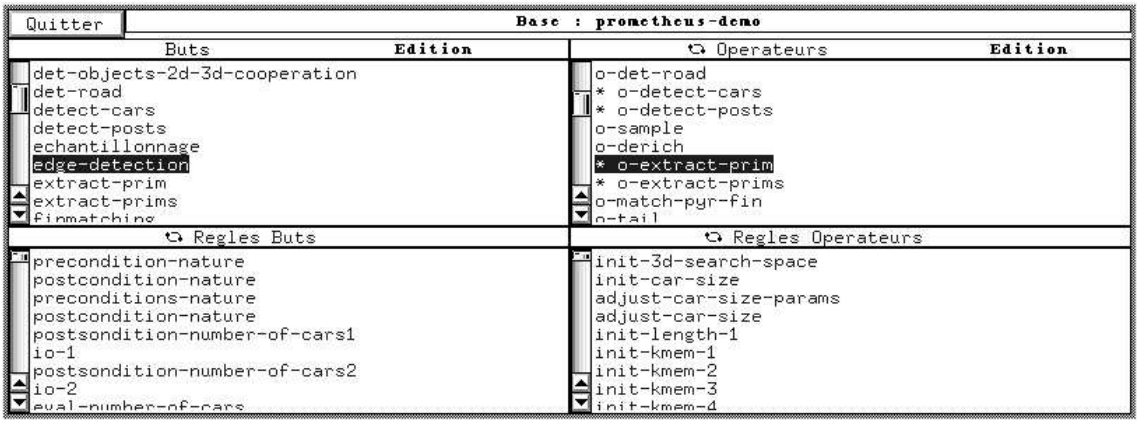


FIG. 2 – *Visualisation de la base*

La figure 3, montre deux extraits d'une base de connaissance sous forme arborescente ; le premier arbre représente la décomposition de l'opérateur *o-extract-prims*, le second celle du but *chainage*. Chaque but, opérateur ou requête de la base de connaissance peut ainsi être visualisé. Il est possible d'éditer cet arbre : soit sa structure, soit des nœuds particuliers. A chaque nœud de l'arbre, un menu est attaché permettant par exemple la visualisation complète de ce nœud par l'intermédiaire de l'éditeur spécialisé. Dans la suite, cet arbre sera appelé arbre statique.

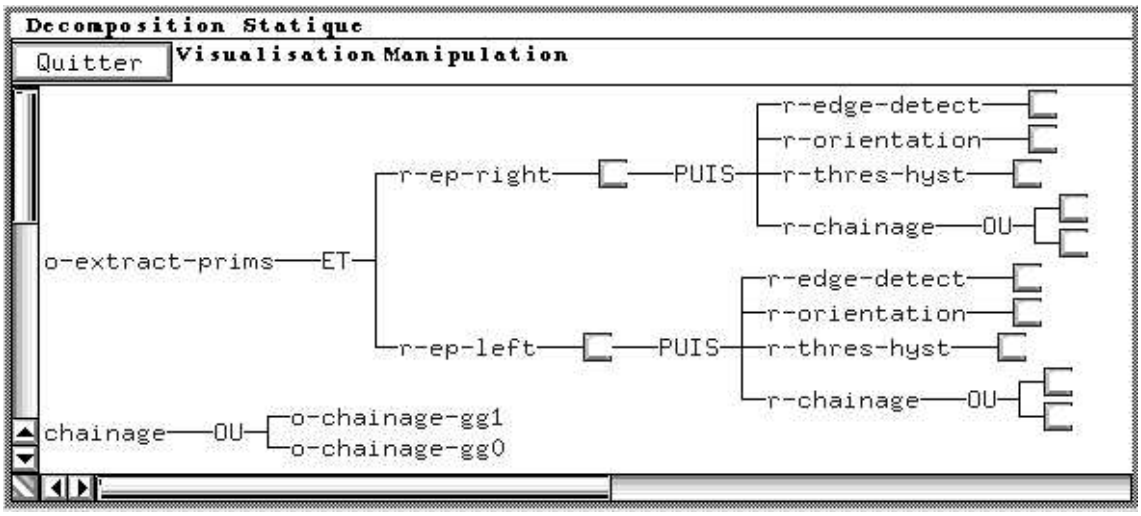


FIG. 3 – *Extraits de la base de connaissance sous forme arborescente*

– Edition des objets par des éditeurs spécialisés (figures 4, 5). Ces éditeurs peuvent être appelés sur les objets présents dans la base de connaissance, et ce à partir de la représentation textuelle ou arborescente de celle-ci. Bien sûr, la création de nouveaux objets se fait par leur intermédiaire.

  Nous présentons figure 4 l'éditeur de but sur le but *consolidation* et l'éditeur d'opérateur sur l'opérateur complexe *o-detect-objects*. La dernière fenêtre, sous l'éditeur de but, montre l'éditeur de paramètres de but.

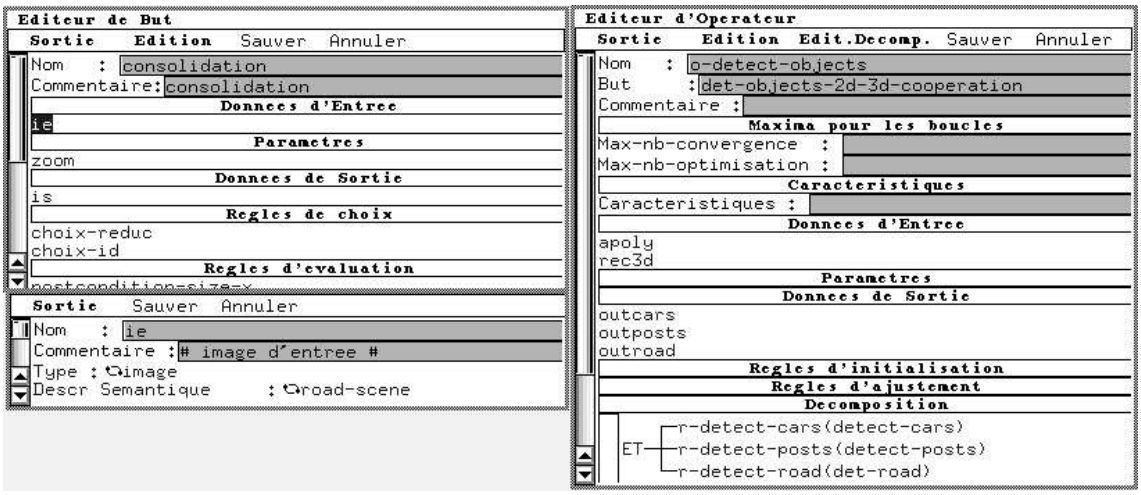


FIG. 4 – *Editeurs de but et d'opérateur*

Les structures des paramètres de buts et d'opérateurs étant différentes, il existe différents éditeurs de paramètres. Nous présentons figure 5 l'édition d'un paramètre d'opérateur.

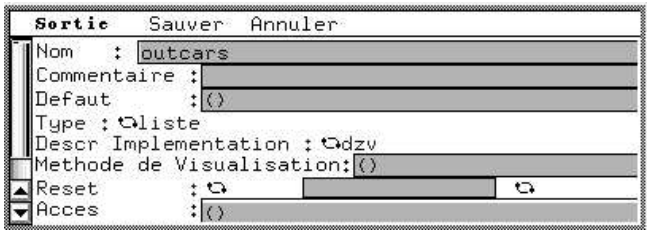


FIG. 5 – *Editeur de paramètre d'opérateur*

– L'éditeur de contexte (figure 6) permet à l'expert de créer une structure de contexte "base de connaissance" et de spécifier les valeurs admissibles pour chacun de ses champs. Ici, nous voyons un élément de contexte simple (*coins*) et deux éléments de contexte structurés (appelés *cameras* et *user-constraints*), qui possèdent des sous-éléments eux-même simples ou structurés.

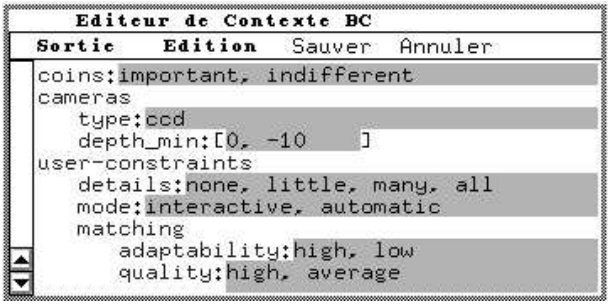


FIG. 6 – *Editeur de contexte BC*

– Edition des règles par un éditeur spécialisé (figure 7).

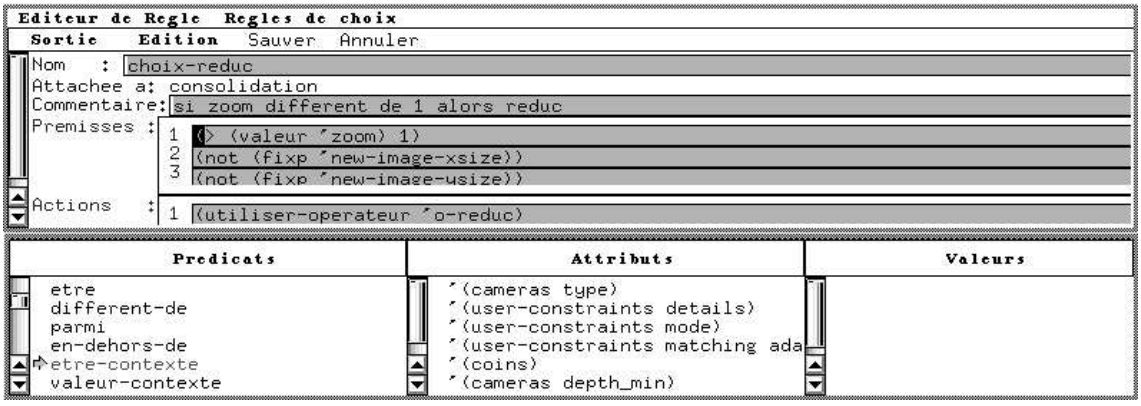


FIG. 7 – *Editeur de règle*

– Edition des descriptions d'implémentation et des descriptions sémantiques associées à chaque type (figure 8).

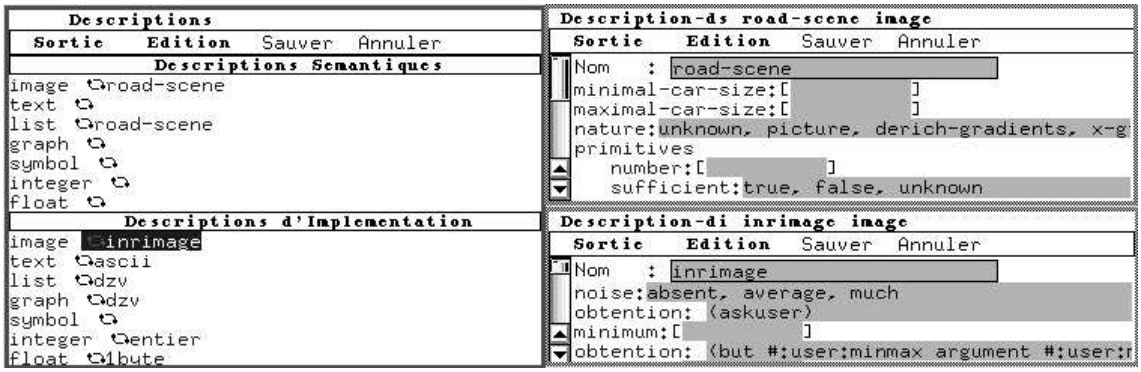


FIG. 8 – *Editeur des descriptions BC*

– Edition des méthodes de visualisation associées à chaque format (figure 9).

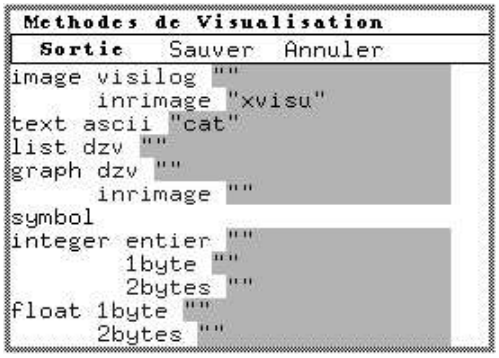


FIG. 9 – *Editeur des méthodes de visualisation*

– Chargement et sauvegarde d'une base de connaissance. Ces fonctions sont appelables directement depuis le menu de développement.

1.2 Menu d'utilisation d'un système expert [Utilisation]

Ce menu s'adresse aussi bien à l'expert en cours de mise au point de sa base de connaissance qu'à l'utilisateur final du système expert.

L'utilisateur dispose d'un éditeur de requêtes ; celui-ci permet de définir une ou plusieurs requêtes qui pourront être soumises au système expert. Dans la figure 10, nous donnons un exemple de requête : il s'agit de la requête nommée *obstacle-ambu* qui travaille entre autres sur deux images : gauche (*s2p1ig*) et droite (*s2p1id*).

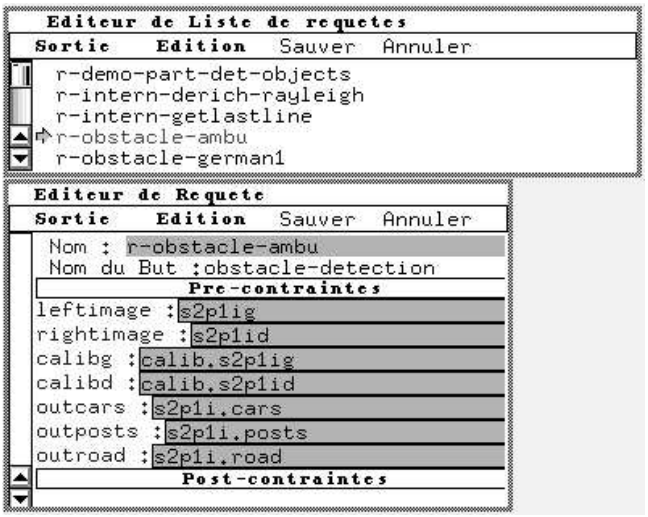


FIG. 10 – *Editeur de requêtes*

Un éditeur de contexte, s'appuyant sur la structure définie par l'expert à l'aide de l'éditeur de contexte Base de Connaissance précédemment illustré (figure 6), est également disponible. Il permet de définir autant de contextes qu'il y a de cas particulier à traiter (figure 11). Il suffit pour cela d'affecter à chaque champ une valeur particulière, symbolique, ou numérique.

Lors du lancement de l'exécution d'une requête, une fenêtre contenant son arbre d'exécution est créée. Elle permet de suivre en temps réel le cheminement du système expert (figure 12) : les nœuds représentent les différentes étapes complexes ; les feuilles sont les programmes exécutés.

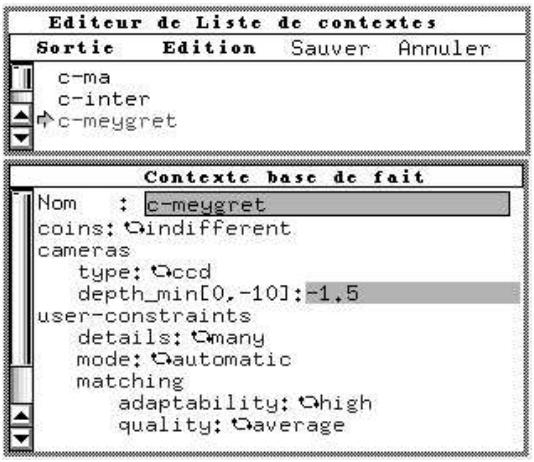


FIG. 11 – *Editeur de contexte utilisateur*

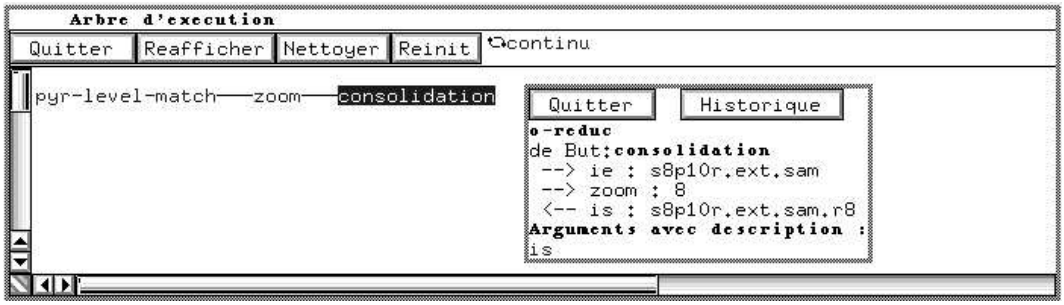


FIG. 12 – *Arbre d'exécution en cours de construction*

Dans la figure 13, nous montrons un arbre d'exécution complet. Il est toujours possible de cliquer (avec la touche Shift ou CTRL enfoncée) sur l'un de ses nœuds pour disposer des informations locales (arguments d'entrée et de sortie).

1.3 Menu de réglage des variables [**Reglage**]

Des menus permettent de positionner des variables de contrôle : visualisation des images intermédiaires en cours de traitement, sauvegarde de la session, etc. (voir figure 14).

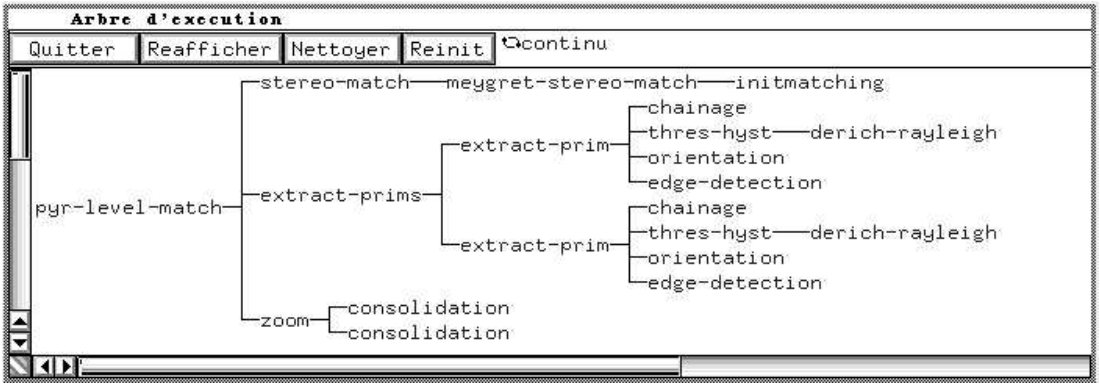


FIG. 13 – *Arbre d'exécution complet*

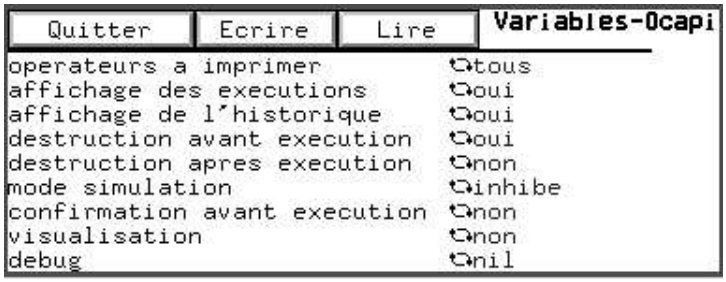


FIG. 14 – *Un des tableaux de variables de contrôle*

2 Menus et dispositifs de cliquage de l'interface

2.1 Préliminaires

Nous décrivons les différents objets graphiques utilisés dans l'interface et la manière d'utiliser le dispositif de cliquage (souris) pour effectuer les actions associées à ces objets graphiques.

2.2 Les Objets Graphiques de l'interface OCAPI

Barres de Défilement

C'est un objet graphique qui permet de représenter dans une fenêtre de taille fixe une liste d'objets généralement représentés par leur nom. Si la liste dépasse la taille de la fenêtre, certains objets ne seront pas visibles mais grâce à un ascenceur il est possible de déplacer la zone de visibilité de la liste d'objets : les déplacements importants se font en déplaçant à la souris le rectangle mobile (ou curseur), les déplacements plus fins en cliquant sur l'une des deux flèches.

Menus

Un menu est composé d'une liste fixe d'items. Lorsque le menu est activé, un seul de ces items peut être sélectionné. L'action associée à cet item est alors déclenchée. Dans la suite, les noms de menus seront indiquées entre crochets [ ], et les noms d'item entre chevrons < >.

Boutons

C'est le dispositif le plus simple pour activer une action ; il suffit d'appuyer sur le bouton. Dans la suite, les noms de boutons seront entourés par un rectangle ▭.

Bouton radio

C'est un bouton à deux positions : appuyé ou relaché. Il sert à représenter l'état d'une variable.

Objet Sélectionnable

C'est un objet que l'on peut sélectionner parmi une liste d'autres objets. Lorsqu'il est sélectionné, il apparaît en grisé (ou d'une couleur différente de celle des autres objets) sur l'écran.

Editeur de ligne

C'est un outil graphique permettant d'éditer une ligne de caractères en utilisant les mêmes clés que l'éditeur Emacs. Il faut cliquer sur la ligne, à l'endroit où on veut écrire pour rendre l'éditeur actif. Le curseur est symbolisé par une ligne verticale ou un pavé noir. Un Retour Chariot est **obligatoire** pour enregistrer les modifications apportées au texte dans la structure interne Lisp associée à l'éditeur.

Arbre

C'est la représentation sous forme arborescente d'un objet complexe constitué de sous-objets. Deux menus sont associés à un arbre :

- un menu global,
- un menu associé aux nœuds.

Les actions associées au menu global sont effectuées sur tous les nœuds de l'arbre ; celles associées au menu local sont déclenchées sur tous les nœuds du sous-arbre où a été sélectionné le menu.

Barillet

C'est un dispositif permettant de faire apparaître successivement les éléments d'une liste d'objets ou de valeurs. Certains barillets déclenchent une action à chaque fois que l'on change d'objet ou de valeur. Dans la suite, un barillet sera représenté entre parenthèses ( ).

2.3 Comment utiliser la souris

D'une manière générale, le bouton gauche et le bouton droit permettent de sélectionner un objet soit dans une barre de défilement soit dans un menu ; le bouton du milieu est réservé au déclenchement des actions associées aux objets sélectionnables.

Parcourir une barre de défilement

En déplacant la souris positionner le curseur dans le triangle supérieur (resp.inférieur), puis cliquer avec le bouton gauche ou droit.

Sélectionner un menu

Placer le curseur sur le libellé du menu, appuyer sur le bouton gauche ou droit puis tirer afin de sélectionner l'item désiré ; relacher alors le bouton : l'action associée à cet item se déclenche.

Appuyer sur un bouton

Placer le curseur sur le bouton et appuyer sur un des boutons de la souris.

Sélectionner un objet sélectionnable

Placer le curseur sur cet objet et appuyer sur le bouton gauche ou droit de la souris.
Lorsqu'une action est associée à l'objet sélectionnable, on sélectionne l'objet et on déclenche l'action associée en appuyant sur le bouton du milieu.

Sélectionner le nœud d'un arbre

Cliquer avec le bouton gauche ou droit sur ce nœud, cela signifie que le nœud sélectionné devient le nœud courant de l'arbre. Le nœud courant d'un arbre est celui à partir duquel sont effectués les ajouts de nouveaux nœuds.

Sélectionner un menu associé à chaque nœud d'un arbre

Appuyer sur la touche Control (ou Shift) et cliquer simultanément avec l'un des boutons de la souris, puis agir comme avec un menu normal.

Faire tourner un barillet

Cliquer avec le bouton du milieu sur l'icône.

3 Comment démarrer OCAPI?

3.1 L'environnement

Attention : toute base créée sous l'environnement anglais ne peut être rechargée et utilisée que sous l'environnement anglais, de même pour l'environnement français.

Lors de la première utilisation d'OCAPI, un fichier *$HOME/.ocapi* sera créé automatiquement par le système pour y stocker les références aux bases de connaissances qui seront créées. Dans la figure 15, deux bases de connaissances existent : *prometheus-demo* et *sesam*.

```
(setq #:ocapi:kb-paths
  (list
     "/u/lyre/orion/ocapi/exemples/prometheus-demo.kb"
     "/u/lyre/orion/ocapi/exemples/sesam.kb"))
```

FIG. 15 – *Un exemple de fichier $HOME/.ocapi*

Chacun des fichiers à suffixe *.kb* contient la description d'une base de connaissances. La figure 16 montre le contenu du fichier *sesam.kb*. Ces informations sont celles qui sont éditées à l'aide de l'éditeur de bases de connaissances (voir détails page 30).

```
(bcgp "sesam"
       "gestion du code de structure interne cesam "
       "Obs. Meudon"
       "0."
       "/u/lyre/0/orion/ocapi/exemples/sesam/"
       '(sesam.ll)
       '(mass-age-star mass-age-approx ltmt domain-det near-init-mod cre-bin
compilation create-prog mod-init-det init-agemax initialisation
init-mass cre-cesam-don cre-script-ascii cre-script-bin run-cesam
init-evol dfm test-dat xy-ini nuc-ini essai essai1 rename-dat)
)
```

FIG. 16 – *Un exemple de fichier à suffixe* .kb

Dans le répertoire indiqué en cinquième argument (qui doit **impérativement** exister et se terminer par /), ici "/u/lyre/0/orion/ocapi/exemples/sesam/", trois répertoires de noms respectifs *bc, bf* et *lisp* doivent exister (les deux premiers sont créés automatiquement par OCAPI sinon). Dans le premier, *bc*, seront stockés tous les objets de la base de connaissances. Dans le deuxième, *bf*, sera stocké tout ce qui concerne les exécutions. Lors du lancement de l'exécution du système expert,

OCAPI se positionne dans le répertoire *bf.* Dans le troisième, *lisp*, l'expert stockera les fichiers Le_Lisp contenant les définitions de variables et de fonctions dont il peut avoir besoin (dans certaines règles, par exemple) et qui seront chargés en même temps que la base de connaissances dont ils dépendent.

Il est également conseillé de créer au même endroit un répertoire *bin* dans lequel l'expert mettra les commandes de lancement (exécutables ou scripts) correspondant aux opérateurs élémentaires décrits dans la base de connaissances ainsi que ses propres outils de visualisation. Nous rappelons qu'**avant** d'appeler *ocapi* il lui faudra alors mettre à jour sa variable unix *PATH* en y indiquant ce répertoire.

3.2 Création d'une nouvelle base

L'ordre des opérations décrites dans cette section doit être **scrupuleusement** respecté.

Lancer OCAPI

Lancer le gestionnaire de multi-fenêtres X11. Appeler dans une fenêtre xterm la commande : ocapi *taille.* L'argument *taille* est optionnel ; il indique la taille de la zone des listes de l'environnement Lisp ; il doit être au moins égal à celui pris en compte pour générer le core du système. La bannière OCAPI et le tableau de bord composé du bouton : **Quitter Ocapi** et des menus : [**Developpement**], [**Utilisation**] et [**Reglage**] apparaissent.

Appeler l'éditeur d'entêtes de bases de connaissances

Cliquer sur l'item <**Editer Bases** > du menu [**Developpement**]. L'éditeur de liste de bases apparaît et sa liste est vide.
Voir les détails dans la section **Editeur d'entêtes de bases de connaissances** page 30.

Créer une base

Cliquer sur l'item <**Nouvel Objet**> du menu [**Edition**]. Un éditeur de base s'affiche avec une base de nom *XXX*, dont tous les champs sont vides. L'utilisateur doit alors donner un nom à sa base, et compléter les autres champs. Ces informations seront stockées dans le fichier <nom-de-base>.kb référencé dans le fichier $HOME/.ocapi. Pour enregistrer, cliquer sur l'item <**Sauver et Quitter**> du menu [**Sortie**].

Sauver la définition de la base de connaissances

Cliquer sur l'item <**Sauver BC sur fichiers**> du menu [**Developpement**].

Lorsque la zone de dialogue apparaît, cliquer sur la base à sauvegarder, puis cliquer sur la case **Entête** pour ne sauvegarder que l'entête. Puis cliquer sur le bouton **D'accord**.

Déclarer la base de connaissances comme base courante

Pour qu'OCAPI prenne en compte cette nouvelle base comme étant la base courante (celle que l'on pourra éditer, modifier, étendre ou faire exécuter), il faut maintenant la charger par l'item <**Charger BC depuis fichiers**> du menu [**Developpement**]. Il est maintenant possible de créer des buts et autres objets dans cette base de connaissances.

Créer un nouveau but

Pour créer un but, cliquer sur l'item <**Visualiser Base**> du menu [**Developpement**] . Une fenêtre avec 4 sous-fenêtres apparaît alors. Cliquer sur l'item <**Nouveau But**> du menu [**Edition**] de la sous-fenêtre **Buts** . Editer le nom, le commentaire concernant ce nouveau but, etc. Cliquer l'item <**Sauver et Quitter**> du menu [**Sortie**] de l'éditeur de but. Créer ensuite les opérateurs éventuels associés à ce but et les sauver (cf page **??**).

Créer un contexte BC

Cliquer sur l'item <**Edit Contexte BC**> du menu [**Developpement**] et suivre les indications données page pagerefecbc.

Sauvegarder la base créée

Cliquer sur l'item <**Sauver BC sur fichiers**> du menu [**Developpement**]. Lorsque la zone de dialogue apparaît, cliquer sur la base à sauvegarder, puis éventuellement cliquer sur la case **Buts** , pour ne sauvegarder que certains buts (il faut ensuite selectionner les buts à sauver) et enfin cliquer sur le bouton **D'accord**.

Charger la base créée

Cliquer dans le menu [**Developpement**] sur l'item <**Charger BC depuis fichiers**> et Lorsque la zone de dialogue apparaît, cliquer sur la base à charger, puis cliquer sur le bouton **D'accord**.

Créer une requête utilisateur

Cliquer dans le menu [**Utilisation**] sur l'item <**Editer Requêtes** >, l'éditeur de liste de requêtes apparaît, il faut alors cliquer sur l'item <**Nouvel Objet**> du menu [**Edition**] , éditer la requête, puis sauvegarder et fermer l'éditeur.

Créer un contexte utilisateur

Cliquer dans le menu [**Utilisation**] sur l'item <**Editer Contextes** >, l'éditeur de liste de contextes apparaît, il faut alors cliquer sur l'item <**Nouvel Objet**> du menu [**Edition**] , éditer le contexte, puis sauvegarder et fermer l'éditeur.

Lancer l'exécution

Cliquer dans le menu [**Utilisation**] sur l'item <**Lancer**>, choisir une requête initiale et un contexte utilisateur.

Sortir d'Ocapi

Cliquer sur le bouton **Quitter Ocapi**.

4 Comment modifier une base de connaissances ?

Entre deux sessions, une base de connaissances (si elle a été sauvegardée) est conservée sous le répertoire *bc*, dans les fichiers suivants :

- des fichiers de règles (-reg.ll);
- des fichiers d'objets buts et opérateurs (-okp.ll).

Les actions de sauvegarde dans les éditeurs d'OCAPI (boutons **Sauver et Quitter** et **Sauver**, ne sauvegardent les ajouts et modifications que de manière **locale**, dans l'environnement de travail courant d'une session. La sauvegarde sur fichiers est réalisée par le seul bouton **Sauver BC sur fichiers**, qui permet effectivement de sauver les informations de l'environnement de travail vers ces fichiers.

Pour modifier une base de connaissances, il faut charger la base et utiliser les éditeurs spécialisés. Il faut ensuite sauvegarder les modifications en cliquant sur l'item <**Sauver BC sur fichiers**> du menu [**Developpement**] . Cette action a

pour but de sauvegarder les structures internes modifiées sur des fichiers (-reg.ll et -okp.ll) que l'on pourra ensuite recharger (voir page 29).

Remarque: de la même façon, une base de faits (requêtes initiales et contextes d'utilisation) est sauvegardée sur fichiers entre deux sessions dans le répertoire *bf* (voir page 26).

Charger la base

Cliquer sur l'item <**Charger BC depuis fichiers**> du menu [**Developpement**] .

Visualiser la Base

Cliquer sur l'item <**Visualiser Base**> du menu [**Developpement**].

Appeler les éditeurs spécialisés

On peut appeler les éditeurs de but ou d'opérateur en sélectionnant un but ou un opérateur puis en cliquant l'item <**Editer**> du menu [**Edition**]. On peut également *visualiser* une requête, un but ou un opérateur dans la fenêtre affichant l'arbre statique de décomposition.

Sauvegarder sur fichiers les buts modifiés

L'utilisateur peut sauvegarder sur fichiers l'ensemble des buts en sélectionnant l'item <**Sauver BC sur fichiers**> du menu [**Developpement**]. Il peut également sauvegarder un ou plusieurs but(s) en appuyant sur le bouton **Buts** de la zone de dialogue activée par <**Sauver BC sur fichiers**> puis en sélectionnant le ou les but(s) qu'il désire sauver. Des fichiers sont alors créés ou modifiés dans le répertoire *bc*.

5 Tableau de bord

Nous détaillons ici les 3 menus correspondant au tableau de bord et apparaissant à gauche sous la bannière OCAPI. Il s'agit des menus: [**Developpement**], [**Utilisation**] et [**Reglage**].

5.1 Menu [Developpement]

Ce menu est celui destiné à l'expert pour développer sa base de connaissances.

Les items **<Charger BC depuis fichiers>** , **<Sauver BC sur fichiers>** font apparaître la même zone de dialogue qui permet à l'utilisateur de choisir la base de connaissances qu'il veut charger ou sauver parmi l'ensemble des bases de son environnement. Une case à cocher **Entête** permet de charger ou de sauver la définition de la base. Une case à cocher **Buts** permet de faire apparaître la liste de buts de la base sélectionnée et de choisir le ou les but(s) à charger ou à sauver. Une case à cocher **Contexte** permet de charger ou de sauver le contexte associé à la base. Une case à cocher **Descriptions** permet de charger ou de sauver les descriptions associées à la base.

Voir détails dans la section **Chargement et sauvegarde d'une base de connaissances** page 29.

<Charger BC depuis fichiers>

Permet de charger les fichiers de buts, de règles et les fichiers lisp d'une des bases référencées dans le fichier *$HOME/.ocapi*. La base chargée devient alors la base courante. Il est possible également de charger seulement un ou plusieurs buts de la base courante en cliquant sur le bouton **Buts** puis en sélectionnant le ou les buts à charger.

<Sauver BC sur fichiers>

Permet de sauvegarder sur fichiers les modifications effectuées sur la base.

<Commuter>

Permet de changer de base courante; utilise le même menu que celui proposé pour le chargement d'une base. La nouvelle base sélectionnée est alors chargée en mémoire si elle ne l'a pas été précédemment. Si elle a déjà été chargée, elle n'est pas rechargée depuis les fichiers mais redevient simplement la base courante avec le contenu qu'elle avait en mémoire précédemment.

<Editer Bases >

Appelle l'éditeur d'entêtes de bases de connaissances et y affiche la liste des bases référencées dans le fichier *$HOME/.ocapi*. Permet de mettre à jour certains champs des bases : noms des auteurs, chemin d'accès aux fichiers, version.
Voir détails dans la section **Editeur d'entêtes de bases de connaissances** page 30.

<Editer Contexte BC >

Appelle l'éditeur de contexte de la base de connaissances en cours (la base chargée). Cet éditeur permet d'éditer, d'ajouter ou de supprimer des champs du contexte.
Voir détails dans la section **Editeur de Contexte BC** page 32.

<Editer Descriptions BC >

Appelle l'éditeur de descriptions de la base de connaissances en cours (la base chargée). Cet éditeur permet d'ajouter, de supprimer et de modifier des descriptions.
Voir détails dans la section **Editeur de Descriptions BC** page 33.

<Editer Méthodes de Visualisation >

Appelle l'éditeur des méthodes de visualisation des données de la base de connaissances en cours (la base chargée). Cet éditeur permet d'ajouter, de modifier ou de supprimer des méthodes.
Voir détails dans la section **Méthodes de Visualisation** page 36.

<Visualiser Base>

Fait apparaître un tableau comprenant la liste des buts et la liste des opérateurs de la base courante ainsi que les règles associées à ces buts et opérateurs. Il est alors possible d'accéder aux éditeurs spécialisés de ces objets.
Voir détails dans la section **Visualiser la Base** page 37.

<Arbre Statique>

Permet de *visualiser* une représentation de la base de connaissances sous forme arborescente qui correspond à la décomposition structurelle des buts en opérateurs, eux-mêmes décomposés en sous-requêtes, et à la description des différentes alternatives permettant de réaliser un but.
Voir détails dans la section **Arbre de Décomposition Statique** page 51.

<Imprimer base>

Permet de sortir sur fichiers, sous une forme externe lisible le contenu de la base de connaissances courante : tous les buts et les opérateurs ainsi que leurs règles associées. Le résultat obtenu est un ensemble de fichiers (un par but de la base), dans le répertoire *bc* de la base de connaissances courante, chacun ayant pour nom : *nom-but.text*.

<Imprimer but>

Permet de de sortir sur fichier, sous une forme claire et lisible un but particulier, tous les opérateurs qui lui sont associés, ainsi que toutes les règles du but et des opérateurs. Le fichier de sortie s'appele toujours : *nom-but.text* et se trouve dans le répertoire *bc* de la base de connaissances courante.

5.2 Menu [Utilisation]

Ce menu s'adresse à l'utilisateur du système expert.

<Lancer>

Lance l'exécution de la base courante. Une zone de dialogue apparaît. Il faut choisir une requête parmi celles qui sont proposées, puis choisir un contexte parmi ceux qui sont proposés.
Une fenêtre s'affiche alors, elle permet de visualiser l'exécution en faisant apparaître dynamiquement les buts et les opérateurs sous forme arborescente.

<Editer Requêtes >

Appelle l'éditeur de requêtes. La liste des requêtes est celle de la base courante.
Voir détails dans la section **Editeur de requêtes** page 57.

<Editer Contextes >

Appelle l'éditeur de contextes. La liste des contextes est celle de la base courante.
Voir détails dans la section **Editeur de Contextes Base de Faits** page 59.

<Sauver Base de Faits sur fichiers>

Sauvegarde sur fichier la liste des requêtes ainsi que la liste des contextes de la base de faits concernant la base courante. Le fichier créé se trouve dans le répertoire *bf* et porte le nom *nom-base.bf*.

5.3 Menu [Reglage]

Ce menu s'adresse principalement à l'expert en cours de test sur sa base de connaissances.

Les éditeurs associés aux items **<Variables Ocapi>**, **<Variables Lisp>**, **<Flux Ocapi>** et **<Buts à visualiser>** permettent de modifier des variables de contrôle d'OCAPI. Ils fonctionnent de la manière suivante : à l'appel de l'éditeur, c'est la valeur actuelle de la variable qui est montrée ; il est possible en cliquant sur les barillets de dérouler les différentes valeurs que peuvent prendre ces variables ; une fois que de nouvelles valeurs ont été choisies pour une ou plusieurs des variables, il est alors **nécessaire** d'appuyer sur le bouton **ECRIRE** pour valider la(les) valeur(s) ainsi sélectionnée(s). Le bouton **LIRE** permet de voir les valeurs actuelles des variables. Le bouton **STOP** permet de fermer l'éditeur.

<Variables Ocapi>

Appelle un éditeur qui permet de positionner des variables Ocapi.
Une copie d'écran de cet éditeur est montrée page 11 figure 14.

<Variables Lisp>

Appelle un éditeur qui permet de positionner des variables Lisp.

<Flux Ocapi>

Appelle un éditeur qui permet de positionner des variables de flux de sortie pouvant être soit dirigées sur l'écran, soit redirigées vers des fichiers (de noms prédéfinis).

<Buts à visualiser>

Appelle un éditeur qui permet de positionner pour chacun des buts de la base de connaissances courante un indicateur de visualisation des arguments des opérateurs exécutés. A l'initialisation (chargement ou commutation de base), tous ces indicateurs sont mis à *non*. Quelles que soient leurs valeurs, si, dans l'éditeur présenté par l'item **<Variables Ocapi>** la visualisation n'a pas été demandée, aucune visualisation ne sera faite en cours d'exécution. Si, dans cet éditeur, la visualisation a été demandée une visualisation sera proposée pour tous les opérateurs correspondant à un but pour lequel l'indicateur de visualisation sera à *oui*.

Si l'expert veut une fois pour toutes choisir des valeurs pour les variables, plutôt que d'avoir à les spécifier à chaque session, il est possible de les fixer dans le fichier *$HOME/.ocapirc* . Voici les commandes à mettre dans ce fichier pour cela, ainsi

que les valeurs correspondantes, telles qu'elles sont indiquées dans les barillets de l'interface :

Commande	Valeurs possibles	Visualisation interface	Valeurs
Variables Ocapi :			
(defvar #:gp:op-exe-print	'tous)	operateurs a imprimer	tous
	nil)		aucun
	'elementaires)		elementaires
	'complexes)		complexes
(defvar #:gp:comline-print	t)	affichage des executions	oui
	nil)		non
(defvar #:gp:historique-print	t)	affichage de l'historique	oui
	nil)		non
(defvar #:gp:destruction-avant-exe	t)	destruction avant execution	oui
	nil)		non
(defvar #:gp:destruction-apres-exe	t)	reset apres execution	oui
	nil)		non
(defvar #:gp:simulation	'inhibe)	mode simulation	inhibe
	'actif)		actif
(defvar #:gp:visualisation	t)	visualisation	oui
	nil)		non
(defvar #:gp:confirme-avant-comline	t)	confirmation avant execution	oui
	nil)		non
(defvar #:gp:debug	t)	debug	oui
	nil)		non

Commande	Valeurs possibles	Visualisation interface	Valeurs
Variables Lisp :			
(setq #:system:redef-flag	t)	redef-flag	t
	nil)		nil
(setq #:system:read-case-flag	t)	read-case-flag	t
	nil)		nil
(setq #:system:print-case-flag	t)	print-case-flag	t
	nil)		nil
(setq #:system:print-for-read	t)	print-for-read	t
	nil)		nil

Par exemple la ligne "(defvar #:gp:confirme-avant-comline nil)", dans le fichier *$HOME/.ocapirc*, permettra de ne pas avoir à donner une confirmation avant chaque exécution.

Bouton **Quitter Ocapi**

Permet de terminer une session, de quitter à la fois Ocapi et Lisp. L'utilisateur se retrouve sous le système d'exploitation.

6 Chargement et sauvegarde d'une base de connaissances

La représentation de la base de connaissances par un ensemble de fichiers Unix est nécessaire pour pouvoir travailler d'une session à l'autre. Les fichiers Unix utilisés pour la sauvegarde d'une base de connaissances sont organisés de la manière suivante :

- Un fichier *nom-base.kb* est destiné à représenter l'entête de la base. Le chemin Unix où se situe ce fichier est donné dans le fichier *.ocapi*, lui-même situé dans le répertoire principal (Home Directory).
- Un répertoire *bc* dans lequel sont stockés, pour chaque but, un fichier *nom-but-okp.ll* et un fichier *nom-but-reg.ll*. Ces fichiers contiennent la représentation externe du but considéré, des opérateurs associés ainsi que de l'ensemble des règles. Ce répertoire contient également un fichier *nom-base-ctxte.ll* dans lequel est stockée la représentation externe du contexte de la base ainsi qu'un fichier *nom-base-descr.ll* dans lequel sont stockés la représentation externe des descriptions associées aux types des données de la base et les méthodes de visualisation associées aux formats des données.
- Un répertoire *bf* qui représente la base de faits. Il contient un fichier *nom-base.bf* où sont stockés les noms de l'ensemble des requêtes ainsi que les noms de l'ensemble des contexte de la base de faits. Il contient en outre un fichier *nom-base-ctxtes.ll* qui représente l'ensemble des contextes et, pour chaque requête, un fichier *nom-requête-okp.ll*.

La zone de dialogue qui apparaît lorsque l'on veut charger ou sauvegarder une base de connaissances est la même (voir figure 17).

Les cases à cocher **Entête**, **Contexte**, **Buts** et **Descriptions** peuvent ou non être cochées. Toute case cochée signifie que l'item associé sera chargé ou sauvegardé. Toutefois, si aucune case n'est cochée, l'ensemble de la base est chargée ou sauvegardée. La base qui est sauvegardée **doit être la base courante**, cependant il est

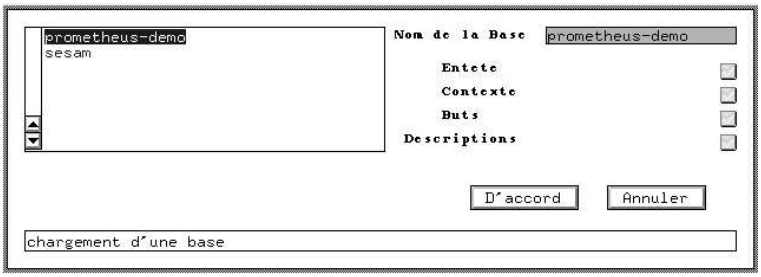


FIG. 17 – *Sauvegarde et chargement d'une base de connaissances*

possible de sauvegarder l'entête d'une autre base que la base courante. Le fait de cocher sur la case **Buts** a pour effet de visualiser la liste des buts de la base précédemment sélectionnée. En sélectionnant ensuite un ou plusieurs de ces buts, on peut ainsi charger ou sauvegarder un sous-ensemble de buts d'une base. Il faut cependant que la base soit la base courante.

7 Editeur d'entêtes de bases de connaissances

Cet éditeur est appelé par l'item <**Editer Bases** > du menu [**Developpement**]. Un exemple est montré page 3 figure 1.

L'éditeur d'entêtes de base de connaissances est un éditeur de liste qui permet de visualiser, modifier, créer ou supprimer une base de connaissances.

Les objets traités par cet éditeur de liste sont des descriptifs de bases de connaissances, aussi appelés *entête*.

L'entête d'une base de connaissances comprend les champs suivants :

- **Nom :** le nom de la base de connaissances.
- **Désignation :** une désignation plus explicite de la base de connaissances qui apparait dans la bannière lorsque la base est chargée.
- **Auteurs :** le (ou les) auteurs de la base.
- **Version :** le numéro de version de la base.
- **Chemin :** le chemin du répertoire Unix où sont stockés les fichiers de la base de connaissances. Ce chemin doit se terminer par /.

– **FichsLisp :** une liste de noms de fichiers Lisp séparés par des blancs.

7.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur d'entête et met à jour le fichier *$HOME/.ocapi* si une base a été introduite ou supprimée, et met à jour les fichiers *base.kb* de description de chaque base pour chacune des entêtes qui a été modifiée.

<Quitter>

Ferme l'éditeur d'entête de base sans sauvegarde.

7.2 Menu [Edition]

<Editer>

Affiche l'entête de la base sélectionnée dans l'éditeur d'entêtes.
Les champs *Nom, Désignation, Auteurs, Version, Chemin, FichsLisp* sont modifiables. A chacun est associé un éditeur de ligne.

<Nouvel Objet>

Ajoute une base à la liste des bases, et affiche son entête dans l'éditeur. L'utilisateur doit fournir les éléments nécessaires à la description de la base.

<Couper>

Supprime la base sélectionnée de la liste des bases et la garde en mémoire jusqu'à la sortie de l'éditeur, ou jusqu'à ce qu'une autre base ait été coupée ou copiée.
<Couper> est utilisé pour supprimer une base ou pour la déplacer dans la liste.

<Copier>

Mémorise la base sélectionnée jusqu'à la sortie de l'éditeur, ou jusqu'à ce qu'une autre base ait été coupée ou copiée.
<Copier> est utilisé pour dupliquer ou déplacer une base dans la liste.

<Coller>

Permet d'insérer dans la liste des bases une base précédemment coupée ou copiée. Si aucune base n'est sélectionnée, l'insertion se fait au début de la liste, sinon elle se fait après la base sélectionnée.

8 Editeur du contexte BC

Cet éditeur peut être appelé depuis l'item **<Edit Contexte BC>** du menu [**Developpement**]. Un exemple est montré page 7 figure 6.

Le contexte de la base de connaissances est constitué d'un ensemble d'éléments de contexte qui peuvent être structurés (à plusieurs niveaux) ou simples. Un élément simple comprend un nom, un type (symbolique ou numérique), un domaine (les valeurs possibles pour un symbolique, les bornes de l'intervalle pour un numérique) et sa caractéristique (information globale ou locale). Un élément de contexte structuré peut être lui-même composé d'éléments de contexte simples ou structurés. L'éditeur de contexte permet de visualiser et de modifier les champs du contexte ainsi que l'ensemble des valeurs pour chacun des champs. Attention : aucun contrôle de cohérence n'est effectué pour l'instant. Pour de plus amples informations se reporter au document ([MVT+95]).

8.1 Menu [Sortie]

<Sauver et Quitter>

Permet de quitter l'éditeur de contexte BC. Le contexte est sauvegardé en mémoire.

<Quitter>

Permet de quitter l'éditeur de contexte BC sans sauvegarde.

8.2 Menu [Edition]

<Nouvel Elt Simple>

Si un élément structuré a été au préalable sélectionné, le nouvel élément simple lui est ajouté comme fils, sinon l'élément est ajouté en fin. Une zone de dialogue demande à l'utilisateur le nom du nouvel élément simple, ainsi que son type (symbolique, numérique) et sa caractéristique (champ global ou local). Deux boutons permettent soit de confirmer soit d'annuler l'affectation de cet ensemble de valeurs à l'élément nouvellement créé. Par défaut le type est symbolique et la caractéristique globale.

<Nouvel Elt Struct>

Si un élément structuré a été au préalable sélectionné, le nouvel élément structuré lui est ajouté comme fils, sinon l'élément est ajouté en fin. Une zone de dialogue demande à l'utilisateur le nom du nouvel élément structuré. Deux boutons permettent soit de confirmer soit d'annuler la création de cet élément.

<Detruire Element Description>

Supprime l'élément sélectionné. Si l'élément est un élément structuré, tous les sous-éléments s'y rapportant sont aussi supprimés.

8.3 Les boutons Sauver et Annuler

Le bouton **Sauver** permet d'enregistrer dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

Le bouton **Annuler** supprime les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

9 Editeur des descriptions BC

Cet éditeur peut être appelé depuis l'item **<Edit Descriptions BC>** du menu [**Developpement**]. Un exemple est montré page 8 figure 8.

Les descriptions permettent de décrire les données. Les descriptions sont attachées aux types prédéfinis dans OCAPI (*image, courbe, liste, symbolique, entier, flottant* et *texte*). Une description se décompose en deux parties : une description

d'implémentation et une description sémantique. Pour de plus amples informations se reporter au Manuel d'OCAPI.

9.1 Menu [Sortie]

<Sauver et Quitter>

Permet de quitter l'éditeur de descriptions. Sauvegarde les descriptions en mémoire.

<Quitter>

Permet de quitter l'éditeur de descriptions sans sauvegarde.

9.2 Menu [Edition]

<Créer Description>

Permet de créer une nouvelle description (sémantique ou d'implémentation) associée à un type. Une zone de dialogue demande à l'utilisateur le nom de la nouvelle description. Une fenêtre spécifique pour cette description apparaît alors (voir la section **Editeur de Descriptions** page 35).

<Editer Description >

Permet de visualiser et de modifier une description sémantique ou d'implémentation. Une fenêtre spécifique pour cette description apparaît alors.

<Supprimer Description>

Permet de détruire une description sémantique ou d'implémentation.

9.3 Les boutons Sauver et Annuler

Le bouton **Sauver** permet d'enregistrer dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde

Le bouton **Annuler** supprime les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

10 Editeur de descriptions

La fenêtre engendrée par les items <**Créer Description**> et <**Editer Description** > permet de visualiser et de modifier une description. Une description est composée d'éléments de description simples ou structurés (à plusieurs niveaux). Un élément simple de description sémantique comprend un nom, un type (symbolique, numérique) et un domaine (les valeurs possibles pour un symbolique, les bornes de l'intervalle pour un numérique). Un élément simple de description d'implémentation comprend en outre la définition d'un moyen d'obtention (soit une forme Lisp, soit une référence à un but déjà défini dans la base et à son argument de retour). Pour de plus amples informations se reporter au Manuel d'OCAPI.

10.1 Menu [Sortie]

<Sauver et Quitter>

Permet de quitter cet éditeur et de retourner à l'éditeur appelant. Sauvegarde en mémoire les modifications apportées à la description.

<Quitter>

Permet de quitter cet éditeur et de retourner à l'éditeur appelant sans sauvegarde.

10.2 Menu [Edition]

<Nouvel Elt Simple>

Si un élément structuré a été au préalable sélectionné, le nouvel élément simple lui est ajouté comme fils, sinon l'élément est ajouté en fin. Une zone de dialogue demande à l'utilisateur le nom du nouvel élément simple, ainsi que son type (symbolique, numérique). Deux boutons permettent soit de sauver soit d'annuler l'affectation de cet ensemble de valeurs à l'élément nouvellement créé. Par défaut le type est symbolique.

<Nouvel Elt Struct>

Si un élément structuré a été au préalable sélectionné, le nouvel élément structuré lui est ajouté comme fils, sinon l'élément est ajouté en fin. Une zone de dialogue demande à l'utilisateur le nom du nouvel élément structuré. Deux boutons permettent soit de sauver soit d'annuler la création de cet élément.

<Detruire Element Description>

Supprime l'élément sélectionné. Si c'est un élément structuré supprime tous les sous-élément s'y rapportant.

10.3 Les boutons Sauver et Annuler

Le bouton **Sauver** permet d'enregistrer dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

Le bouton **Annuler** supprime les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

11 Editeur des méthodes de visualisation

Cet éditeur peut être appelé depuis l'item <**Editer Méthodes de Visualisation** > du menu [**Developpement**] . Un exemple est montré page 8 figure 9. Les méthodes de visualisation sont spécifiques à chaque format. Un éditeur ligne apparaît à côté de chaque format précédemment déclaré.

<Sauver et Quitter>

Permet de quitter l'éditeur de méthodes. Sauvegarde les méthodes en mémoire.

<Quitter>

Permet de quitter l'éditeur de méthodes sans sauvegarde.

11.1 Les boutons Sauver et Annuler

Le bouton **Sauver** permet d'enregistrer dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

Le bouton **Annuler** supprime les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

12 Visualiser la base

La fenêtre engendrée par l'item <**Visualiser Base**> du menu [**Developpement**] est composée de quatre sous-fenêtres (un exemple est montré page 4 figure 2) :

- sous-fenêtre **Buts :** la liste alphabétique des noms de buts de la base,
- sous-fenêtre **Opérateurs :** la liste des noms d'opérateurs de la base,
- sous-fenêtre **Règles Buts :** la liste des règles associées aux buts de la base,
- sous-fenêtre **Règles Opérateurs :** la liste des règles associées aux opérateurs de la base.

Deux types d'actions sont possibles à partir de cette fenêtre :

- des actions de visualisation concernant les buts, les opérateurs et les règles,
- des actions d'édition qui font appel à des éditeurs spécialisés : éditeur de but, éditeur d'opérateur et éditeur de règle.

12.1 Les actions de visualisation

Ce sont les actions qui permettent de voir les liens de dépendance entre les différents objets de la base de connaissances (Buts, Opérateurs et Règles). Elles sont accessibles en cliquant directement sur un nom d'objet visible dans la fenêtre engendrée par l'item <**Visualiser Base**>. D'une manière générale, le fait de cliquer (avec le bouton du milieu) sur un objet permet de visualiser les objets qui lui sont dépendants : par exemple, cliquer sur un but montre dans la sous-fenêtre **Règles Buts**, les règles qui lui sont attachées, dans la sous-fenêtre **Opérateurs**, les opérateurs qui lui sont attachés, dans la sous-fenêtre **Règles Opérateurs**, les règles associées aux opérateurs attachés au but.

D'autre part, des barillets sont attachés à trois de ces fenêtres :

- sous-fenêtre **Opérateurs :** (**Opérateurs...**) contenant les options Opérateurs, Opérateurs Complexes et Opérateurs Elémentaires.

- sous-fenêtre **Règles Opérateurs :** (**Règles Opérateurs...**) contenant les options Règles Opérateurs, Règles d'initialisation et Règles d'ajustement.
- sous-fenêtre **Règles Buts :** (**Règles Buts...**) contenant les options Règles Buts, Règles de choix et Règles d'évaluation.

Ils vont permettre de filtrer les objets de la sous-fenêtre correspondante, en ne réalisant que la visualisation des objets qui correspondent à l'option choisie dans le barillet. Dans les sections suivantes, lorsque nous parlerons de cliquage sur un barillet ce sera sur l'icône, et de cliquage sur un bouton, de même nom que le barillet, ce sera sur le texte situé juste à droite de l'icône.

Cliquer sur un but avec le bouton du milieu

Affiche les noms des opérateurs et des règles associées suivant l'option choisie dans les barillets correspondants.

Cliquer sur un opérateur avec le bouton du milieu

Affiche le nom du but et des règles associées suivant l'option choisie dans les barillets correspondants.

Cliquer sur le barillet (Règles Buts...) avec le bouton du milieu

Change la valeur du barillet et effectue l'action associée au bouton dont le titre est affiché. Cette action est l'affichage des noms de toutes les règles des buts, ou seulement des règles de choix ou seulement des règles d'évaluation.

Cliquer sur le barillet (Opérateurs...) avec le bouton du milieu

Change la valeur du barillet et effectue l'action associée au bouton dont le titre est affiché. Cette action est l'affichage des noms de tous les opérateurs, ou seulement des opérateurs complexes ou seulement des opérateurs élémentaires.

Cliquer sur le barillet (Règles Opérateurs...) avec le bouton du milieu

Change la valeur du barillet et effectue l'action associée au bouton dont le titre est affiché. Cette action est l'affichage des noms de toutes les règles, ou seulement des règles d'ajustement ou seulement des règles d'initialisation.

Cliquer sur le bouton Buts

Affiche la liste de tous les noms de buts de la base courante, et suivant la position des barillets des autres sous-fenêtres affiche la liste des noms d'opérateurs, la liste des règles associées aux buts, la liste des règles associées aux opérateurs.

Cliquer sur le bouton Règles Buts

Réaffiche la sélection de règles correspondante.

Cliquer sur le bouton Opérateurs

Réaffiche la sélection de noms d'opérateurs correspondante.

Cliquer sur le bouton Règles Opérateurs

Réaffiche la sélection de règles correspondante.

12.2 Edition des Buts

Il s'agit ici du menu [**Edition**] proposé dans la partie droite du bandeau de la sous-fenêtre **Buts**.

<Editer>

Si un but est sélectionné, ouvre l'éditeur de but s'il ne l'est pas déjà, et y affiche la description de ce but.

<Nouveau But>

Crée un nouveau but dans la base, ouvre l'éditeur de but et y affiche la description d'un but vide de nom XXX.
Si un but est sélectionné, le nom du nouveau but s'insère dans la liste des buts juste après celui-ci, sinon il est inséré en fin de liste.

<Supprimer But>

Supprime le but sélectionné de la base.

12.3 Edition des Opérateurs

Il s'agit ici du menu [**Edition**] proposé dans la partie droite du bandeau de la sous-fenêtre **Opérateurs**.

<Editer>

Si un opérateur est sélectionné, ouvre l'éditeur d'opérateur s'il ne l'est pas déjà, et y affiche la description de cet opérateur.

<Nouvel Opérateur>

Crée un nouvel opérateur dans la base, ouvre l'éditeur d'opérateur et y affiche la description d'un opérateur vide de nom XXX.
Une zone de dialogue demande à l'utilisateur s'il s'agit d'un opérateur élémentaire ou complexe.
Si un opérateur est sélectionné, le nom du nouvel opérateur s'insère dans la liste des opérateurs juste après celui-ci, sinon il est inséré en fin de liste.

<Supprimer Opérateur>

Supprime l'opérateur sélectionné de la base.

12.4 Edition des Règles

Pour appeler l'éditeur de règle, il faut sélectionner la règle à partir de l'éditeur du but ou de l'opérateur auquel elle est rattachée, puis sélectionner dans le menu [**Edition**] l'item <**Editer champ**>, voir la section **Editeur de Règle** page 47.

13 Editeur de But

L'éditeur de but peut être appelé depuis la fenêtre de visualisation de la base (engendrée par l'item <**Visualiser Base**> du menu [**Developpement**]. Un exemple est montré page 6 figure 4.

Les champs du but sont alors affichés dans une fenêtre et peuvent éventuellement être modifiés. Le dernier champ : *Opérateurs* affiche les opérateurs dont le champ fonction est le même que celui du but édité ; ce champ n'est pas modifiable, il est mis à jour automatiquement, par exemple lorsqu'un opérateur associé au but est introduit.

13.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur de but et ses sous-éditeurs éventuels (éditeurs d'argument, de règle,...). Sauvegarde en mémoire les modifications.

<Quitter>

Ferme l'éditeur de but et ses sous-éditeurs éventuels (éditeurs d'argument, de règle,...) sans sauvegarde.

13.2 Menu [Edition]

<Editer Champ>

Si un argument (paramètre, donnée d'entrée ou de sortie) ou une règle est sélectionné, ouvre l'éditeur de l'argument ou de la règle, s'il ne l'est pas déjà, et y affiche la description de cet argument ou de cette règle (voir **Editeur d'argument de but** page 49 et **Editeur de règle** page 47).

<Nouvelle Donnée d'Entrée>

Ajoute une donnée d'entrée au but, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom XXX. La nouvelle donnée d'entrée est insérée dans la liste des données d'entrée du but, à la fin de la liste ou après la donnée qui est sélectionnée.

<Nouvelle Donnée de sortie>

Ajoute une donnée de sortie au but, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom XXX. La nouvelle donnée de sortie est insérée dans la liste des données de sortie du but, à la fin de la liste ou après la donnée qui est sélectionnée.

<Nouveau Paramètre>

Ajoute un paramètre au but, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom XXX. Le nouveau paramètre est inséré dans la liste des paramètres du but, à la fin de la liste ou après le paramètre qui est sélectionné.

<Supprimer Argument>

Supprime l'argument qui a été sélectionné.

<Nouvelle Règle de Choix>

Crée une nouvelle règle de choix de nom XXX et appelle un éditeur de règle. La nouvelle règle est insérée dans la liste des règles de choix du but à la fin de la liste ou après la règle qui est sélectionnée.

<Nouvelle Règle d'Evaluation>

Crée une nouvelle règle d'évaluation de nom XXX et appelle un éditeur de règle. La nouvelle règle est insérée dans la liste des règles d'évaluation du but à la fin de la liste ou après la règle qui est sélectionnée.

<Supprimer Règle>

Supprime la règle qui est sélectionnée dans l'éditeur de but.

14 Editeur d'opérateur

L'éditeur d'opérateur peut être appelé depuis la fenêtre de visualisation de la base courante (qui est engendrée par l'item **<Visualiser Base>** du menu [**Developpement**]). Un exemple est montré page 6 figure 4.

Les champs de l'opérateur sont alors affichés dans une fenêtre et peuvent éventuellement être modifiés.

Le nom de l'opérateur à la création est *XXX* puis doit être modifié par l'utilisateur.

Le nom du but est un champ important puisque quand l'utilisateur le choisit, l'opérateur nouvellement créé prend les mêmes arguments que le but auquel il est associé.

Le dernier champ est différent selon qu'il s'agit d'un opérateur complexe ou élémentaire.

- Dans le cas d'un opérateur élémentaire, il s'agit du champ **appel** dont le sous-champ *langage* est choisi en utilisant un barillet, et le sous-champ *syntaxe* à l'aide d'un éditeur de ligne. On utilisera en général pour un appel une commande shell ou Lisp. Dans ce cas le barillet est positionné sur *shell* ou *lisp* et dans le champ *syntaxe* apparaît la ligne de commande entre parenthèses.

- Dans le cas d'un opérateur complexe, il s'agit du champ **Décomposition** qui correspond à un menu ([**Edit.Decomp.**]), décrit page 45.

14.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur d'opérateur et ses sous-éditeurs éventuels (éditeurs de paramètre, de règle,...). Sauvegarde en mémoire les modifications.

<Quitter>

Ferme l'éditeur d'opérateur et ses sous-éditeurs éventuels (éditeurs de paramètre, de règle,..) sans sauvegarde.

14.2 Menu [Edition]

<Editer Champ >

Si un paramètre, une donnée d'entrée ou de sortie, ou une règle est sélectionné, ouvre l'éditeur de l'argument ou de la règle, s'il ne l'est déjà, et y affiche la description de cet argument (voir **Editeur d'argument d'opérateur** page 50 et **Editeur de règle** page 47).

<Nouvelle Donnée d'Entrée>

Ajoute une donnée d'entrée de l'opérateur, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom *XXX*. La nouvelle donnée d'entrée est insérée dans la liste des données d'entrée de l'opérateur, à la fin de la liste ou après la donnée qui a été sélectionnée.

<Nouvelle Donnée de sortie>

Ajoute une donnée de sortie de l'opérateur, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom *XXX*. La nouvelle donnée de sortie est insérée dans la liste des données de sortie de l'opérateur, à la fin de la liste ou après la donnée qui a été sélectionnée.

<Nouveau Paramètre>

Ajoute un paramètre de l'opérateur, ouvre l'éditeur d'argument et y affiche la description d'un argument vide de nom *XXX*. Le nouveau paramètre est inséré dans la liste des paramètres de l'opérateur, à la fin de la liste ou après le paramètre qui a été sélectionné.

<Supprimer Argument>

Supprime l'argument qui a été sélectionné.

<Nouvelle Règle d'Initialisation>

Crée une nouvelle règle d'initialisation de nom XXX et appelle un éditeur de règle. La nouvelle règle est insérée dans la liste des règles d'initialisation de l'opérateur, à la fin de la liste ou après la règle d'initialisation sélectionnée.

<Nouvelle Règle d'Ajustement>

Crée une nouvelle règle d'ajustement de nom XXX et appelle un éditeur de règle. La nouvelle règle est insérée dans la liste des règles d'ajustement de l'opérateur, à la fin de la liste ou après la règle d'ajustement sélectionnée.

<Supprimer Règle>

Supprime la règle qui est sélectionnée dans l'éditeur d'opérateur.

14.3 Menu [Edit.Decomp.]

Il s'agit du cas d'un opérateur complexe ; ce menu permet d'éditer l'arbre de décomposition de l'opérateur.

<Couper sous-arbre>

Supprime le nœud sélectionné et l'ensemble de ses descendants.
Supprime l'arbre en entier si aucun nœud n'a été sélectionné.

<Ajouter Requete apres courante>

Ajoute un nouveau nœud dans l'éditeur d'arbre, comme frère du nœud sélectionné. Ceci permet d'indiquer l'ordre souhaité des requêtes.
Une zone de dialogue permet de guider l'utilisateur dans le choix du nouveau nœud à ajouter en fonction de sa position dans l'arbre.
Le nœud à ajouter dans ce cas est une requête, la zone de dialogue comporte deux ascenceurs et deux éditeurs de ligne pour éditer le nom de la requête et le nom du but correspondant.

<Ajouter Requete fille (1ere)>

Ajoute un nouveau nœud dans l'éditeur d'arbre, comme premier fils du nœud racine (un ajout sur un autre nœud est impossible pour le moment).
Si le nœud à ajouter est une liste de requêtes (nœud *Puis* ou *Et*), la zone de dialogue est plus complexe, elle est décrite figure 18.
L'utilisateur doit choisir un nom de requête dans l'éditeur de ligne de gauche, puis un nom de but associé en sélectionnant ce nom dans l'ascenceur de droite. Il repètera cette opération autant de fois qu'il souhaite de requêtes. La liste des requêtes s'affiche dans l'ascenceur de gauche. Pour supprimer une requête ainsi créée, il suffit de cliquer sur le nom de cette requête dans l'ascenceur de gauche et de cliquer sur **Annuler**.

FIG. 18 – *Zone de dialogue pour créer une liste de requêtes*

14.4 Menu associé aux nœuds de l'arbre de décomposition.

La sélection du menu associé aux nœuds se fait en appuyant sur la touche Control (ou Shift) et en cliquant sur le nœud avec l'un quelconque des boutons de la souris.

<Editer>

Permet d'appeler l'éditeur correspondant au nœud sélectionné. En l'occurence, il s'agit d'éditeurs de requêtes (cf figure 19). Les différents éditeurs appelés se superposent et leur gestion est à la charge de l'utilisateur.

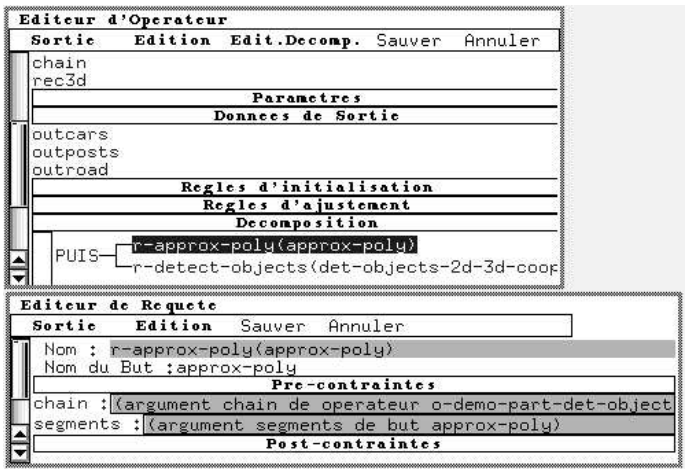


FIG. 19 – *Editeur d'opérateur et éditeur de requête associé*

<Reafficher>

Réaffiche le sous-arbre dont la racine est le nœud sélectionné, en réarrangeant de manière symétrique l'ensemble des fils de chaque nœud.

<Flot de données>

Affiche le flot de données entre les requêtes de l'arbre de décomposition.

15 Editeur de Règle

Cet éditeur permet de visualiser et de modifier une règle. Un exemple est montré page 7 figure 7. Les champs d'une règle sont les suivants :

- **Nom :** le nom de la règle;
- **Commentaire :** le commentaire de l'expert sur la règle;
- **Attachee a :** le nom du but ou de l'opérateur auquel est attachée la règle (non éditable);
- **Prémisses :** l'ensemble des prémisses de la règle, numérotées ;
- **Actions :** l'ensemble des actions de la règle numérotées.

15.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur de règle. Sauvegarde en mémoire les modifications (arguments, règles, arbre de décomposition, ...).

<Quitter>

Ferme l'éditeur de règle sans sauvegarde.

15.2 Menu [Edition]

<Nouvelle Prémisse>

Ajoute une prémisse à la liste des prémisses de la règle. Un nouveau numéro de prémisse apparaît suivi d'un éditeur qui peut être utilisé pour éditer la prémisse. Une aide est fournie sous forme de plusieurs ascenceurs (voir Remarque page 48) quand on sélectionne le bouton **Prémisses** et le numéro de cette prémisse. Pour que la prémisse soit prise en compte, il faut appuyer sur le bouton **Sauver** ou taper Retour Chariot.

<Nouvelle Action>

Ajoute une nouvelle action dans la liste des actions de la règle. Un nouveau numéro d'action apparaît suivi d'un éditeur qui peut être utilisé pour éditer l'action. Une aide est fournie sous forme de plusieurs ascenceurs (voir Remarque page 48) quand on sélectionne le bouton **Actions** et le numéro de cette action. Pour que l'action soit prise en compte, il faut appuyer sur le bouton **Sauver** ou taper Retour Chariot.

<Couper>

Supprime la prémisse ou l'action sélectionnée. Pour sélectionner une prémisse ou une action, cliquer sur **Prémisses** (resp. **Actions**) puis sur le numéro de la prémisse (resp. de l'action).

15.3 Aide à l'édition de règle

Lors de la création d'une nouvelle prémisse ou d'une nouvelle action, une aide est fournie à l'utilisateur. Cette aide est matérialisée par trois ascenceurs contenant des listes de symboles que l'on peut insérer dans l'éditeur de ligne sélectionné en cliquant simplement sur le symbole que l'on désire insérer (voir page 7 figure 7).
Le premier ascenceur intitulé **Prédicats** contient les noms des prédicats utilisables dans une prémisse ou dans une action selon le cas. Lorsque l'on clique sur l'un des prédicats proposés, le deuxième ascenceur affiche un ensemble de symboles qui dépend de la sélection.
Le deuxième ascenceur intitulé **Attributs** contient les noms des arguments dans le cas d'une prémisse et le nom des arguments des opérateurs dans le cas d'une action

ou encore le nom d'un champ du contexte de la base de connaissances. Cliquer sur un des symboles de cette liste, insère ce symbole dans l'éditeur de ligne et affiche éventuellement des valeurs dans le troisième ascenceur.
Le troisième ascenceur intitulé **Valeurs** contient les valeurs du champs du contexte courant qui a été sélectionné dans le deuxième ascenceur.

16 Editeur d'argument de but

Les différents champs visualisés dans cet éditeur sont les suivants :

Nom :	le nom du paramètre s'édite grâce à un éditeur de ligne.
Commentaire :	C'est un commentaire indiquant la fonction de l'argument.
Type :	le type de l'argument est entier, réel, image, texte, liste, courbe ou symbole. Un barillet permet de sélectionner le type (bouton du milieu sur l'icône représentant le barillet). Le type par défaut est image.
Description sémantique :	c'est le nom de description sémantique associée au type choisi. Un barillet permet de sélectionner la description sémantique (bouton du milieu sur l'icône représentant le barillet).

Bouton **Sauver**

Enregistre dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

Bouton **Annuler**

Annule les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

16.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur de champ et enregistre les modifications dans la base.

<Quitter>

Ferme l'éditeur de champ sans sauvegarde.

17 Editeur d'argument d'opérateur

Les différents champs visualisés dans cet éditeur sont les suivants :

Nom : le nom de l'argument s'édite grâce à un éditeur de ligne.

Commentaire : C'est un commentaire indiquant la fonction du paramètre qui s'édite grâce à un éditeur de ligne.

Défaut : la valeur par défaut du paramètre. C'est une valeur numérique, symbolique ou une expression. Si c'est une valeur numérique ou un symbole, il suffit d'écrire la valeur ou le symbole dans l'éditeur de ligne. Si c'est une expression, il faut la mettre entre parenthèses.

Type : le type de l'argument est soit entier, réel, image, texte, liste, courbe, ou symbolique. Un barillet permet de sélectionner le type (bouton du milieu sur l'icône représentant le barillet).

Descr Implementation : permet d'indiquer quelle description d'implémentation (ou format), parmi celles définies dans la BC, on veut affecter à cet argument (par exemple : inrimage, ascii, dzv, ...). Un barillet permet de sélectionner le format (bouton du milieu sur l'icône représentant le barillet).

Méthode de Visualisation : permet d'exprimer une méthode de visualisation spécifique au champ, elle est plus prioritaire que celle associée au format. Ce champ est de type chaine de caractères et doit correspondre au nom d'un exécutable de visualisation pour ce type d'argument existant dans votre système.

Reset : un barillet permet de sélectionner les différentes valeurs possibles pour ce champ (*avant* ou *après* qui indique si un reset (destruction des données) doit être fait et quand, c'est-à-dire si cela doit être avant l'exécution de l'opérateur (problèmes de compatibilité de formats) ou après son exécution (gestion de place mémoire). Un éditeur de ligne permet de spécifier par un symbole comment faire la testruction des données(par exemple: *rmfile*). Enfin, un autre barrillet permet de choisir si cette destruction est impérative ou facultative.

Accès : Ce champ est utilisé dans le cas d'un argument de sortie dont la valeur se trouve dans un fichier produit par le programme exécuté par l'opérateur. Il s'agit d'indiquer où se trouve cette valeur dans le fichier. La syntaxe utilisée est la suivante: $ITEM < entier > [DE\_LA\_LIGNE\{< entier > | < symbole >\}] DU\ FICHIER < expression >$ où [] désigne une option et | désigne une alternative.

Intervalle : pour les paramètre numériques on peut ici indiquer un intervalle de variation possible de la valeur du paramètre.

Si l'opérateur est élémentaire et que l'argument est un argument de sortie, un menu [**Edition**] apparaît dans le bandeau de l'éditeur. L'item <**Editer Moyen d'Obtention**> de ce menu [**Edition**] permet de redéfinir les moyens d'obtention des champs de la description d'implémentation associée.

Bouton **Sauver**

Enregistre dans la base les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

Bouton **Annuler**

Supprime les modifications effectuées depuis l'appel de l'éditeur ou depuis la dernière sauvegarde.

17.1 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur de champ et enregistre les modifications dans la base.

<Quitter>

Ferme l'éditeur de champ sans sauvegarde.

18 Arbre de décomposition statique

On peut visualiser la base de connaissances sous une forme arborescente qui correspond à la décomposition structurelle des buts en opérateurs, eux-mêmes décomposés en sous-buts (sous-requêtes) et à la description des différentes alternatives permettant de réaliser un but.

Un exemple est montré page 5 figure 3.

Bouton **Quitter**

Ferme la fenêtre de visualisation de la décomposition des buts et requêtes.

18.1 Menu [Visualisation]

La visualisation des opérateurs, des requêtes et des buts permet de montrer sous forme arborescente la décomposition structurelle de la base de connaissances suivant l'axe horizontal (décomposition des buts en sous-buts) ; ainsi que les différentes alternatives de réalisation d'un but par des opérateurs distincts.

<Décomposer Opérateur>

Demande dans une zone de dialogue quel opérateur doit être décomposé, puis affiche l'arbre de décomposition complet de cet opérateur.

<Décomposer Requête>

Demande dans une zone de dialogue quelle requête doit être décomposée, puis affiche l'arbre de décomposition de cette requête.

<Décomposer But>

Demande dans une zone de dialogue quel but doit être décomposé puis affiche l'arbre de décomposition de ce but.

18.2 Menu [Manipulation]

<Mini Arbre>

Affiche un arbre en réduction, où les noms des nœuds n'apparaissent pas. Mini Arbre est utilisé principalement pour les gros arbres, pour mieux visualiser l'ensemble de l'arbre et permettre des recherches de nœuds. L'arbre réduit est le sous-arbre dont la racine est le nœud sélectionné.

<Libellé>

Si le nœud sélectionné est affiché sous forme réduite, affiche le nom complet de ce nœud.

<Libellés>

Lorsque l'arbre est affiché sous forme réduite (mini-arbre), demande l'affichage de tous les nœuds du sous-arbre du nœud sélectionné.

<Libellés Ops>

Lorsque l'arbre est affiché sous forme réduite (mini-arbre), demande l'affichage de tous les nœuds Opérateur du sous-arbre du nœud sélectionné.

<Nettoyer>

Supprime, dans le sous-arbre dont la racine est le nœud d'où est lancée la commande, l'aspect grisé de certains nœuds s'ils existent.

<Reafficher>

Réaffiche le sous-arbre dont la racine est le nœud d'où est lancée la commande en réarrangeant de manière symétrique l'ensemble des fils de chaque nœud.

<Rechercher Op>

Demande, dans un zone de dialogue, de sélectionner un nom d'opérateur parmi l'ensemble des opérateurs du sous-arbre dont la racine est le nœud depuis lequel on lance la recherche.
Recherche, dans ce sous-arbre, un ou plusieurs opérateurs dont le nom a été sélectionné. Le ou les nœuds trouvés apparaissent alors en grisé.

<Rechercher But>

Demande, dans une zone de dialogue, de sélectionner un nom de but parmi l'ensemble des buts du sous-arbre dont la racine est le nœud depuis lequel on lance la recherche.
Recherche, dans ce sous-arbre, un ou plusieurs buts dont le nom a été sélectionné. Le ou les nœuds trouvés apparaissent alors en grisé.

<Rechercher Req>

Demande, dans une zone de dialogue, de sélectionner un nom de requête parmi l'ensemble des requêtes du sous-arbre dont la racine est le nœud depuis lequel on lance la recherche.
Recherche, dans ce sous-arbre, une ou plusieurs requêtes dont le nom a été sélectionné. Le ou les nœuds trouvés apparaissent alors en grisé.

18.3 Menu associé aux nœuds de l'arbre statique

Ce menu est pratiquement identique au précédent, cependant un item a été ajouté : <**Visualiser**>.
Nous ne décrivons pas les items qui apparaissent dans le menu précédent, ils ont la même signification à la différence qu'ils s'appliquent au sous-arbre d'où est lancée la commande.

<**Visualiser**>

Permet d'appeler l'éditeur correspondant au nœud sélectionné, en mode visualisation seulement. Les différents éditeurs appelés se superposent et leur gestion est à la charge de l'utilisateur. Toutefois la fermeture de l'éditeur d'arbre provoque la fermeture de tous ces éditeurs.

19 Arbre d'exécution

Cet arbre d'exécution apparaît dans une fenêtre lorsque la commande *Lancer* est appelée. Il a pour but de visualiser l'exécution en cours en affichant les noms des buts et sous-buts qui sont exécutés. Des exemples sont montrés page 10 figure 12 et page 11 figure 13.

Chaque but est représenté par un nœud de l'arbre et est éventuellement suivi par la succession de ses sous-buts qui sont représentés par les fils de ce nœud.

Quand un but est réalisé par un opérateur *op1* puis par un autre opérateur *op2* au cours de la même exécution, le sous-arbre d'exécution associé à *op1* disparaît pour laisser la place au sous-arbre d'exécution de *op2*. L'utilisateur peut cependant visualiser l'historique de l'exécution à l'aide du menu associé aux nœuds de l'arbre.

Attention : quand en cours d'exécution un nœud est sélectionné (clic de gauche ou du milieu), il devient alors le nœud courant ; c'est-à-dire que l'affichage des nœuds exécutés ensuite sera réalisé à partir du nœud sélectionné. Ainsi la structure de l'arbre d'exécution ne sera pas cohérente avec l'exécution. Pour visualiser le contenu d'un nœud, faire donc très attention à utiliser un clic avec Control (ou Shift) pour faire apparaître le menu associé.

19.1 Les boutons

Les boutons sont situés en haut à gauche de la fenêtre de l'arbre d'exécution.

Bouton **Quitter**

Ferme la fenêtre de l'arbre d'exécution ainsi que les fenêtres associées à cet arbre. **Attention :** une fois la fenêtre quittée, il ne sera plus possible d'avoir accès à cette exécution ou à son historique.

Bouton **Reafficher**

Réaffiche l'arbre ou les arbres contenus dans la fenêtre en réarrangeant de manière symétrique l'ensemble des fils de chaque nœud.

Bouton **Nettoyer**

Ferme les fenêtres associées à l'arbre d'exécution. Ces fenêtres sont les fenêtres d'affichage des opérateurs et de l'historique.

Bouton **Reinit**

Ferme la fenêtre de l'arbre d'exécution et en ouvre une nouvelle. **Attention :** une fois la fenêtre fermée, il ne sera plus possible d'avoir accès à cette exécution ou à son historique.

19.2 Le mode d'utilisation du barillet.

A droite des boutons, se trouve un barillet qui règle le mode d'utilisation. L'utilisateur a la possibilité de contrôler l'exécution en modifiant ce barillet qui comporte trois positions :

continu : c'est le mode par défaut, le contrôle est laissé au système.

pas-a-pas : dans ce mode, l'utilisateur a la possibilité d'arrêter ou de poursuivre l'exécution à chaque fois qu'une nouvelle requête est lancée. Une fenêtre de dialogue permet de demander à l'utilisateur l'arrêt ou la poursuite de l'exécution. Si on choisit l'arrêt, la fenêtre de l'arbre d'exécution n'est pas fermée automatiquement ; on garde donc accès aux informations contenues dans l'arbre d'exécution pour la partie de traitement qui a effectivement été exécutée.

arrêt-choix : la possibilité d'arrêter ou de poursuivre l'exécution est alors offerte à chaque fois que le système effectue un choix entre plusieurs alternatives pour réaliser un but.

19.3 Les actions associées aux nœuds de l'arbre d'exécution

<Noms de Buts>

Lorsque l'arbre est affiché sous forme réduite (mini-arbre), demande l'affichage de tous les nœuds du sous-arbre sélectionné.

<Mini Arbre>

Affiche un arbre en réduction, où les noms de nœuds n'apparaissent pas et sont remplacés par des petits rectangles. L'arbre réduit est le sous-arbre du nœud sélectionné.

<Opérateur(s)>

Permet de visualiser dans une (ou plusieurs) fenêtre(s) le (ou les) opérateur(s) qui ont été exécuté(s) successivement pour réaliser le but associé au nœud sélectionné. Un exemple est montré page 10 figure 12.
Chaque fenêtre comprend :

le nom de l'opérateur

les données d'entrée et les paramètres qui sont affichés après une flêche à droite suivie du nom et de la valeur du paramètre et, éventuellement, d'un jugement concernant cette valeur.

les données résultats qui sont affichées après une flêche à gauche suivie du nom et de la valeur du paramètre et, éventuellement, d'un jugement concernant cette valeur.

le jugement global éventuel de l'évaluation de l'opérateur.

le bouton **Quitter** qui ferme la fenêtre de description de l'opérateur.

le bouton Historique : si la réalisation du but a nécessité plusieurs essais, soit avec le même opérateur, soit avec des opérateurs différents, ce bouton permet, pour chaque opérateur de visualiser le sous-arbre d'exécution qui lui est associé. Ce sous-arbre a la même forme et les mêmes fonctionnalités que celui que nous décrivons actuellement.

les descriptions des arguments en sortie de l'opérateur.

Lors de la visualisation d'un opérateur dans une fenêtre spécialisée, les valeurs des paramètres et des données de type numérique ou symbolique sont directement visibles. Par contre, pour les données structurées, c'est simplement leur accès (nom du fichier de stockage) qui est affiché. Pour visualiser les données effectives (contenu des fichiers), cliquer sur le nom du fichier. Un menu apparaît alors, proposant de visualiser les données avec les options par défaut (spécifiques au site d'utilisation ou définies par l'expert) ou bien de choisir vous-même la commande de visualisation à utiliser. Si vous choisissez la deuxième option, un éditeur ligne apparaît qui vous permet de saisir la commande.

20 Editeur de Requêtes

Deux types de requêtes sont à distinguer :

- les requêtes utilisateur qui sont des requêtes initiales et dont la liste est éditée par l'item <**Editer Requête**> du menu [**Utilisation**];
- les requêtes internes aux opérateurs que l'on peut créer et éditer à partir de l'éditeur d'opérateur (menu [**Decomposition**] pour créer une requête et sélection gauche plus Shift pour l'éditer) et que l'on peut aussi visualiser dans l'arbre statique.

20.1 Editeur de liste de requêtes initiales

Cet éditeur de liste est le même que celui de l'éditeur d'entêtes de bases de connaissances. Se reporter donc à cet éditeur pour la signification des menus page 30. Un exemple est montré page 9 figure 10.

Pour créer une nouvelle requête utilisateur, utiliser l'item <**Nouvel Objet**> qui demande le nom du but associé à la requête. La nouvelle requête apparaît alors avec les champs correspondant au but choisi (tels qu'ils sont au moment de la **création** de la requête : si le but change de champs, la requête n'en est pas avertie). Un éditeur de ligne permet à l'utilisateur de donner une valeur à chacun de ces champs (cad de préciser des pré-contraintes et des post-contraintes sur les paramètres ou sur les résultats). Cette valeur peut être un symbole, une chaîne de caractères, un nombre ou plus généralement une expression arithmétique.

Le traitement répétitif a été introduit dans OCAPI pour pouvoir appliquer le même traitement à un ensemble de données. S'il s'agit de données ordonnées, on parle de *séquence*. Pour exprimer une séquence de données, nous avons deux moyens :

- lister toutes les données
- se servir du numéro de la première donnée et de celui de la dernière.

Ainsi quand on veut exprimer une séquence de données comme (`s8p1l.ext, s8p2l.ext`), on peut écrire tout simplement `s8p[1-2]l.ext` dans l'éditeur de requête. Lorsqu'il s'agit d'une séquence longue, cette facilité aide à éviter la fausse frappe (Fig. 20).

De plus on peut indiquer le nom de la description sémantique et celui de la description d'implémentation des arguments en entrée, en sélectionnant un argument, puis en cliquant sur le menu [**Edition**] et l'item <**Editer Descriptions**>. Le système demande de choisir les descriptions et l'utilisateur peut initialiser les champs de ces descriptions dans un éditeur spécialisé.

20.2 Editeur de requête interne

Il s'agit d'un éditeur proche du précédent. L'expression des contraintes est plus riche en ce sens qu'il est possible de faire référence à des paramètres de requêtes issues du même nœud de l'arbre de décomposition ou de l'opérateur ou du but dont la requête est issue.

L'item <**Editer Description** > du menu [**Edition**] permet à l'utilisateur d'initialiser les champs de descriptions **uniquement pour les arguments qui font référence à une valeur**.

Une référence à un argument s'exprime de la manière suivante:
(argument *px* de operateur *o*)
où *px* est le nom d'un des paramètres de l'opérateur *o*. Cette expression peut être

FIG. 20 – *Editeur de requête : exprimer une séquence de données*

incluse dans une expression arithmétique lisp comme par exemple:
(+ (argument *taille* de requete *r2*) (argument *calibration* de operateur *o*))

20.3 Menu [Sortie]

<Sauver et Quitter>

Ferme l'éditeur de requêtes. Sauvegarde en mémoire les modifications.

<Quitter>

Ferme l'éditeur de requêtes sans sauvegarde.

21 Editeur de Contextes Base de Faits

Cet éditeur est un éditeur de listes (voir l'éditeur de liste d'entêtes de bases de connaissances page 30). Un exemple est montré page 10 figure 11.

Pour chaque contexte édité, l'utilisateur positionne la valeur des champs. Si le champ est symbolique ceci se fait en cliquant sur le barillet correspondant avec le bouton du milieu, jusqu'à la valeur souhaitée, si le champ est numérique, l'utilisateur

donne, via un éditeur ligne, une valeur compatible avec les bornes spécifiées par l'expert.

22 Fonctions et variables OCAPI

Quelques fonctionnalités n'ont pas été intégrées de manière graphique dans l'interface et il faut utiliser la fenêtre *Le_Lisp* pour les appeler. Par exemple, la variable *#:ocapi:log* délivre la liste des chemins des fichiers qui ont été modifiés au cours de la session courante.

23 Comment obtenir OCAPI?

Pour obtenir le logiciel OCAPI, contactez :

Projet Orion
INRIA Sophia Antipolis
2004, route des Lucioles
B.P. 93
06902 SOPHIA-ANTIPOLIS Cedex (France)
Tel : (33) 93 65 76 57
Fax : (33) 93 65 79 39

A Lexique Franco-Anglais des termes utilisés

Actions	**Actions**
<Ajouter>	**<New>**
Nouvelle Action	New Action
Nouveau But	New Goal
Nouveau Champ	New Field
Nouvel Elt Simple	New Simple Elt
Nouvel Elt Struct	New Struct Elt
Nouvel Operateur	New Operator
Nouveau Paramètre Entrée	New Input Parameter
Nouveau Parametre Sortie	New Output Parameter
Nouvelle Premisse	New Premisse
Annuler	**Cancel**
Arbre Statique	Static Tree
Arbre d'execution	Execution Tree
arret-choix	stop on choice
Auteurs	Authors
Barillet	Roller
But	Goal
Buts	**Goals**
<Buts a visualiser>	Goals to display
<Charger BC depuis fichiers>	**<Load KB from file>**
Chemin	Path
<Coller>	**<Paste>**
Commentaire	Comment
Contextes	Contexts
continu	non stop
<Copier>	**<Copy>**
<Couper>	**<Cut>**
Nouvelle Description	New Description
Decomposer But	Goal Decomposition
Decomposer Operateur	Operator Decomposition
Decomposer Requete	Request Decomposition
Decomposition	Decomposition
Defaut	Default
Designation	Full Name
Description d'implementation	implementation Description
Description Semantique	Semantic Description

[**Developpement**]	[**Development**]
<Editer>	**<Edit>**
<Editer Base>	**<Edit Base>**
Editer Champ	Edit Field
<Editer Contexte BC>	**<Edit KB Context>**
<Editer contextes>	**<Edit Contexts>**
<Editer Descriptions BC>	**<Edit KB Descriptions>**
<Editer Methodes de Visualisation>	**<Edit Visualisation Methods>**
Editer Paramètre	Edit Field
Editer Règles d'Ajustement	Edit Ajustment Rules
Editer Règles de Choix	Edit Choice Rules
Editer Règles d'évaluation	Edit Evaluation Rules
Editer Règles d'Initialisation	Edit Initialisation Rules
Editer Requête	Edit Request
Editeur de But	Goal Editor
[**Edition**]	[**Edition**]
<Effacer>	**<Delete>**
Effacer	**Delete**
Essai	Attempt
Fonction	Function
Historique	**History**
<Lancer>	**<Run>**
Imprimer base	Print base
Imprimer but	Print goal
Langage	Language
Libelle	Name
Libelles	Names
Libelles Ops	Names and Ops
Nettoyer	Clean-up
Modules	Modules
Noms de Buts	Goal names
<Nouvel Objet>	**<New Object>**
Mini Arbre	Mini Tree
Opérateur	Operator
Operateurs	**Operators**
Operateurs Complexes	**Complex Operators**
Operateurs Elementaires	**Terminal Operators**

pas-a-pas	step by step
Premisses	**Premisses**
Quitter	**Quit**
<Reafficher>	**<Pretty>**
Reafficher	**Pretty**
Recherche But	Search Goal
Recherche Op	Search Op
Recherche Req	Search Request
[Reglage]	**[Tuning]**
Regles Buts	**Goal Rules**
Regles d'ajustement	**Ajustment Rules**
Regles de Choix	**Choice rules**
Regles d'evaluation	**Evaluation Rules**
Regles d'initialisation	**Initialisation Rules**
Regles Operateurs	**Operator Rules**
Requêtes	Requests
Sauver	**Save**
<Sauver et Quitter>	**<Save and Quit>**
<Sauver>	**<Save>**
<Sauver BC sur fichiers>	**<Save KB on files>**
Sauver Contexte	Save Context
Séquence	Sequence
Sortie	Exit
<Supprimer But>	**<Delete Goal>**
<Supprimer Champ>	**<Delete Field>**
<Supprimer Opérateur>	**<Delete Operator>**
<Supprimer Paramètre>	**<Delete Parameter>**
Syntaxe	Syntax
Type	Type
[Utilisation]	**[Run ES]**
Version	Version
Visualisation	Visualisation
<Visualiser Base>	**<Show Base>**

Table des matières

1 Présentation générale **3**
1.1 Menu de développement [**Developpement**] 3
1.2 Menu d'utilisation d'un système expert [**Utilisation**] 9
1.3 Menu de réglage des variables [**Reglage**] 10

2 Menus et dispositifs de cliquage de l'interface **13**
2.1 Préliminaires 13
2.2 Les Objets Graphiques de l'interface OCAPI 13
2.3 Comment utiliser la souris 14

3 Comment démarrer OCAPI? **15**
3.1 L'environnement 15
3.2 Création d'une nouvelle base 17

4 Comment modifier une base de connaissances? **21**

5 Tableau de bord **22**
5.1 Menu [Developpement] 23
5.2 Menu [Utilisation] 25
5.3 Menu [Reglage] 26

6 Chargement et sauvegarde d'une base de connaissances **29**

7 Editeur d'entêtes de bases de connaissances **30**
7.1 Menu [Sortie] 31
7.2 Menu [Edition] 31

8 Editeur du contexte BC **32**
8.1 Menu [Sortie] 32
8.2 Menu [Edition] 33
8.3 Les boutons Sauver et Annuler 33

9 Editeur des descriptions BC **33**
9.1 Menu [Sortie] 34
9.2 Menu [Edition] 34
9.3 Les boutons Sauver et Annuler 34

10 Editeur de descriptions **35**
10.1 Menu [Sortie] 35
10.2 Menu [Edition] 35
10.3 Les boutons Sauver et Annuler 36

11 Editeur des méthodes de visualisation **36**
11.1 Les boutons Sauver et Annuler 36

12 Visualiser la base **37**
12.1 Les actions de visualisation 37
12.2 Edition des Buts 39
12.3 Edition des Opérateurs 40
12.4 Edition des Règles 40

13 Editeur de But **41**
13.1 Menu [Sortie] 41
13.2 Menu [Edition] 41

14 Editeur d'opérateur **42**
14.1 Menu [Sortie] 43
14.2 Menu [Edition] 43
14.3 Menu [Edit.Decomp.] 45
14.4 Menu associé aux nœuds de l'arbre de décomposition. 46

15 Editeur de Règle **47**
15.1 Menu [Sortie] 47
15.2 Menu [Edition] 48
15.3 Aide à l'édition de règle 48

16 Editeur d'argument de but **49**
16.1 Menu [Sortie] 49

17 Editeur d'argument d'opérateur **50**
17.1 Menu [Sortie] 51

18 Arbre de décomposition statique **51**
18.1 Menu [Visualisation] 52
18.2 Menu [Manipulation] 52
18.3 Menu associé aux nœuds de l'arbre statique 54

19 Arbre d'exécution **54**
19.1 Les boutons 54
19.2 Le mode d'utilisation du barillet. 55
19.3 Les actions associées aux nœuds de l'arbre d'exécution 56

20 Editeur de Requêtes **57**
20.1 Editeur de liste de requêtes initiales 57
20.2 Editeur de requête interne 58
20.3 Menu [Sortie] 59

21 Editeur de Contextes Base de Faits **59**

22 Fonctions et variables OCAPI **60**

23 Comment obtenir OCAPI? **60**

A Lexique Franco-Anglais des termes utilisés **61**

Index

.ocapi, 16
.ocapirc, 27
éditer, 14
éditeur
    base, 18

action, 13
Arbre, 14
    menus, 14
arbre
    exécution, 54
Arbre Statique, 25
arrêt-choix, 55
Attributs, 48

barillets, 14
Barres de Défilement, 13
base, 4
    charger, 29
    sauver, 29
    visualisation, *voir* visualiser base
    visualiser, 22
PE chargement, 20
PE charger, 22
PE modifications, 21
bouton, 13
    radio, 13
but
    fenêtre, 20
    rechercher, 53
    sauvegarde, 20
PE creer, 20
PE sauvegarde, 22
Buts, 20, 22, 23
Buts à visualiser, 27

Charger BC, 20, 22, 23
Coller, 32
Commuter, 23
Contexte, 23
contexte, 32
PE créer, 21
continu, 55
Copier, 32
Couper, 31
Créer Description, 34, 35

Décomposer
    But, 52
    Opérateur, 52
    Requête, 52
décomposition, 51
description, 33, 35
Descriptions, 23
Detruire Element Description, 33, 36
Developpement, 3, 17, 22
    Charger BC, 20, 22, 23
    Edit Contexte BC, 32
    Edit Descriptions BC, 33
    Editer Bases, 17, 30
    Editer Méthodes de Visualisation, 36
    Sauver BC, 19–21, 23
    Visualiser Base, 20, 22, 37, 41

Edit Contexte BC, 20, 32
Edit Descriptions BC, 33
Edit.Decomp., 43, 45
Editer, 22, 40
Editer champ, 40
PE Bases, 17, 24, 30

PE Champ, 43
PE Contexte BC, 24
PE Contextes, 21, 26
PE Descripition, 58
PE Description, 34, 35
PE Descriptions BC, 24
PE Méthodes de Visualisation, 24, 36
PE Requête, 26
PE Requêtes, 21
Editeur
    arguments, 50
    Contextes Base de Faits, 59
PE descriptions BC, 8
PE paramètres d'opérateurs, 6
PE requêtes, 9
PE but/opérateurs, 6
Edition, 31
    Editer, 22
    Editer champ, 40
    Editer Descripition, 58
    Nouveau But, 20
    Nouvel Objet, 18, 21
Emacs, 14
Entête, 23
environnement, 15
PE lancer, 21

fichiers
    buts/opérateurs, 21
    configuration, 27
    configurations, 16
    règles, 21
Flot de données, 47
Flux Ocapi, 27

Imprimer base, 25
Imprimer but, 25
items, 13

lancement, 17
Lancer, 25
    Utilisation, 21
Libellé, 52
licence
    OCAPI, 60

Manipulation, 52
menu
    arbre, 54
Menus, 13
PE nœud, 15
Mini Arbre, 52, 56

Nettoyer, 53
Noms de Buts, 56
Nouveau But, 20, 40
Nouveau Paramètre, 42, 44
Nouvel Elt Simple, 33, 35
Nouvel Elt Struct, 33, 36
Nouvel Objet, 18, 21, 31
Nouvel Opérateur, 40
Nouvelle Action, 48
Nouvelle Donnée d'Entrée, 41, 44
Nouvelle Donnée de sortie, 42, 44
Nouvelle Prémisse, 48
Nouvelle Règle d'Ajustement, 44
Nouvelle Règle d'Evaluation, 42
Nouvelle Règle d'Initialisation, 44
Nouvelle Règle de Choix, 42
nœud, 15, 45
nœud
    menus, 15

objet
    sélectionnable, 13
OCAPI
    licence, 60

PE sortir, 21
Opérateur, 50, 56
opérateur
    rechercher, 53
Opérateurs, 37
operateur, 40, 42
Orion
    projet, 60

pas-a-pas, 55
Prédicats, 48
projet Orion, 60

Quitter, 32
Quitter Ocapi, 17, 29

règle, 48
Reafficher, 47, 53
Reglage, 10, 17, 22, 26
repertoires, 17
requête, 57
    initiale, 57
    interne, 58
    rechercher, 53
PE créer, 21

séquence, 58
PE base, 20
Sauver Base de Faits sur fichiers, 26
Sauver BC, 19–21, 23
Sauver et Quitter, 32
souris, 13, 14
Supprimer Argument, 42, 44
Supprimer But, 40
Supprimer Description, 34
Supprimer Opérateur, 40
Supprimer Règle, 42, 45

Utilisation, 9, 17, 21, 22, 25
    Editer Contextes, 21
    Editer Requêtes, 21

Valeurs, 49
Variables Lisp, 27
Variables Ocapi, 26
Visualisation, 52
visualisation, 37
Visualiser Base, 20, 22, 24, 37, 41
visualiser base, 37

Références

[Ala91] C. Alardo. Extensions des facilités de planification du générateur de système expert ocapi. Rapport de dess ia, Université de Paris VI, septembre 1991.

[Clé90] V. Clément. *Raisonnement cognitifs appliques au pilotage d'algorithme de traitement d'images*. PhD thesis, Université de Nice-Sophia Antipolis, 1990.

[CT91] C. Clément and M. Thonnat. Progal : un système expert en traitement d'images de galaxies. In *8me congrès RFIA*, Lyon, Novembre 1991.

[CT93a] C. Clément and M. Thonnat. Pilotage de procédures de traitement d'images pour la description morphologiques de galaxies. *TS*, 9(5), January 1993.

[CT93b] V. Clément and M. Thonnat. Integration of image processing procedures, ocapi : a knowledge-based approach. *CVGIP*, 57(2), March 1993.

[CT93c] V. Clément and M. Thonnat. Pilotage de procédures de traitement d'images pour la description morphologique de galaxies. *TS*, 9(5), January 1993. numéro spécial Intelligence Artificielle.

[Ma90] R. Ma. Extensions du générateur de système expert ocapi. Technical report, I.N.R.I.A., Novembre 1990.

[MPT+95] S. Moisan, F. Ployette, M. Thonnat, V. Clément, and R. Vincent. Documentation de l'interface du logiciel OCAPI version *2.0*. Technical report, I.N.R.I.A. Sophia Antipolis, 1995.

[MVT+95] S. Moisan, R. Vincent, M. Thonnat, V. Clément, and J. van den Elst. Manuel de Référence du logiciel OCAPI version *2.0*. Technical Report 0183, I.N.R.I.A. Sophia Antipolis, 1995.

[TCO95] M. Thonnat, V. Clément, and J. C. Ossola. Automatic galaxy description. *Astrophysical Letters and Communication*, 31(1-6), 1995.

[TCvdE93] M. Thonnat, V. Clément, and J. van den Elst. Supervision of perception tasks for autonomous systems : the ocapi approach. Rapport de Recherche n° 2000, Institut National de Recherche en Informatique et en Automatique, June 1993.

[TCvdE94] M. Thonnat, V. Clément, and J. van den Elst. Supervision of Perception Tasks for Autonomous Systems: the OCAPI Approach. *Journal of Information Science and Technology*, 3(2):140–163, Jan 1994. Also in Rapport de Recherche 2000, 1993, INRIA Sophia Antipolis.

[vdE92] J. van den Elst. Semantical integration of image processing operators for object detection in road scenes. Technical report, I.N.R.I.A and University of Amsterdam, march 1992.

[Vin93] R. Vincent. Conception et réalisation d'un module d'apprentissage pour un générateur de systèmes experts. Rapport de d.e.a., Université de Nice Sophia-Antipolis, septembre 1993.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399